JLC-HDBK-SRAH

March 1997

TECHNICAL REPORT

SOFTWARE REENGINEERING ASSESSMENT HANDBOOK

Version 3.0

This page intentionally left blank.

EXECUTIVE SUMMARY

Existing software is a valuable DoD asset which should be leveraged to the greatest degree possible. There is an immediate need for DoD guidance in conducting technical, economic, and management assessments to determine when software reengineering techniques are cost beneficial.

Such guidance has been developed under the auspices of the Joint Logistics Commanders, Joint Group on Systems Engineering (JLC-JGSE). This Technical Report introduces and details the new *Software Reengineering Assessment Handbook (JLC-HDBK-SRAH) Version 3.0*.

The *SRAH*, formerly titled the *Reengineering Economics Handbook (REH)*, is intended to be used as guidance for conducting an effective technical, economic, and management assessment of existing software to determine whether to maintain, reengineer, or retire that software. Reengineering strategies include: reverse engineering, restructuring, translation, data reengineering, redocumentation, forward engineering, and retargeting.

The decision to reengineer existing software versus maintain or retire that software should be based on standard criteria and a defined process to determine *if*, *when*, and *how* to reengineer. This handbook provides the documentation of a Software Reengineering Assessment (SRA) process including a cost/benefit methodology for DoD. The SRA process is applicable to all types and sizes of software and all levels of an organization, as well as non-DoD, commercial/industrial, and academic organizations, and includes three sequential processes:

1. **Reengineering Technical Assessment (RTA) Process:** The evaluation of candidate software and the selection of reengineering strategies for each candidate using weighted technical criteria and other attributes.

2. **Reengineering Economic Assessment (REA) Process**: The evaluation of the reengineering strategies and the optional selection of preferred strategies using economic indicators.

3. **Reengineering Management Decision (RMD) Process**: The evaluation and rank ordering of candidates and their reengineering strategies using a combination of technical, economic, and other considerations.

The SRA process can be applied to three situations: *Case 1* B Assessing a set of candidates within an organization to determine which has the greatest need to reengineer, and determining if, when, and how to reengineer; *Case 2* B Assessing a specific candidate within the organization; and *Case 3* B Assessing a specific reengineering strategy for a specific candidate.

This handbook is available for use by all departments and agencies of the Department of Defense (DoD). Recommendations for changes, additions, deletions, and/or pertinent data or feedback on its use which may be of use in improving this document should be addressed to the following:

Mr. Robert E. Johnson, Jr.
Single Agency Manager
Architecture and Reengineering
104 Boundary Channel Drive
Arlington, VA 22202-3700
VOICE: 703-693-2740 or DSN 223-2740
FAX: 703-693-2929 or 2888
Email: johnsonro@pentagon-sam.army.mil

Copies of the SRAH Version 3.0 can be downloaded from the USAF/STSC web site at: http://www.stsc.hill.af.mil/

Paper copies can be obtained from the DTIC at 315-334-4933.

CD copies can be obtained from DACS at 315-334-4905.

TABLE OF CONTENTS

APPENDICES

LISTS OF FIGURES AND TABLES

LIST OF FIGURES

FIGURE PAGE

LIST OF FIGURES (CONT.)

LIST OF FIGURES (CONT.)

FIGURE PAGE

LIST OF TABLES

1. Introduction

1.1 Scope

This *Software Reengineering Assessment Handbook (SRAH)* provides information and guidance to individuals responsible for DoD software, specifically in maintaining existing software, implementing software reuse, and incorporating Commercial Off The Shelf (COTS)/ Non-Developmental Items (NDI) into new or existing systems. It focuses on providing practical assistance to the analyst and/or engineer responsible for analyzing existing software for potential reengineering. However, the results of the processes described in this handbook are intended to be used by DoD decision makers as a basis for making informed decisions about whether to maintain, reengineer, or retire existing software. Portions of this process may not be fully applicable to every type of software and may be tailored to each user organization's needs. The following terms are used throughout this handbook:

- **Maintenance –** the continued support of existing software (i.e., status quo).

- **Reengineering –** the examination and alteration of an existing subject software component. This process encompasses a combination of subprocesses such as reverse engineering, restructuring, translating, data reengineering, redocumenting, forward engineering, and retargeting. Emphasis is placed on consolidation, migration, or reuse activities that will lead to shared integrated resources.

- **Retirement –** discontinuing use of a software component.

1.2 Applicability

This handbook applies to the analysis and planning of reengineering efforts for all DoD software. Use of this handbook should be consistent with the provisions of applicable directives. This handbook also applies to other Government and industrial software, (the same principles and processes apply). It provides a defined Software Reengineering Assessment (SRA) process for conducting an effective technical, economic, and management assessment of existing software to determine whether to maintain, reengineer, or retire that software. The technical and economic assessment sub-processes are intended for execution by individuals having a thorough understanding of the existing software, and software economics, or access to someone who has this knowledge. The management decision sub-process is intended for execution by a decision maker familiar with these areas, but not necessarily expert in any of them.

The overall SRA process can be performed at the system level, the Computer Software Configuration Item (CSCI) level for embedded/tactical systems, or the computer program level for Automated Information Systems (AIS). Its applicability ranges from small to very large systems and from a single maintainer to DoD executives responsible for software. The SRA process can be applied at the maintenance organization level, the program office level, or at the systems command level.

The SRA process is applicable to various types of project life cycle approaches including waterfall, incremental, periodic, and spiral. It will be applied more frequently for incremental, periodic, and spiral reengineering, than for waterfall, but on a smaller scale.

The SRA process is designed to be tailored to meet the needs of certain types/classes of software. For example, when tactical (hard real-time, embedded, or weapon-related software) software components are considered, the reengineering decision maker (e.g., Program Manager) may not be choosing between several software components for the best to reengineer. Rather, he/she is faced with a go/no-go choice on reengineering, or is reengineering for other than pure economic reasons and only needs to choose the best available strategy. Thus, some of the handbook processes are not applicable. Furthermore, software engineering trade-off analysis may precede the SRA to address open technical issues. Each user should tailor the SRA process to fit their organization's needs.

1.3 Motivation

Existing software is a valuable DoD asset which must be used to the greatest degree possible. Consequently, there is an immediate need for DoD guidance for conducting technical, economic, and management assessment of reengineering techniques to determine when they are cost beneficial. The *Software Reengineering Executive Strategy and Master Plan* produced at the JLC-JGSE Santa Barbara I Reengineering Workshop (reference 2.1.2e) states:

> The state of DoD software systems is approaching a national crisis. Currently 30% of the software budget is being spent on development, while the other 70% is spent on maintenance. Woefully, 50% of the maintenance resources are expended on understanding system requirements and the design and specifications of existing systems. This catastrophe has drastically affected our war fighting capability and drained our resources. Resources currently devoted to continued maintenance could be better utilized on reengineering or new development with greater return on investment. This alarming trend will be exacerbated unless there is a devoted effort to change the entire process of the system acquisition from cradle to grave. Utilizing systems software reengineering and reuse as a paradigm for capitalizing on our investment in existing and future systems shows great potential for resolving a major portion of the problem ... The vision of this Master Plan is to preserve, extend, and leverage DoD's past and present investments in systems through reengineering for the future. Bold leadership is required to achieve the following goals for the year 2000:

- For existing systems – Reengineer based on return on investment assessment
- For new systems – Develop with reengineering and/or reuse structures
- For technology – Enable working at high levels of aggregation (e.g., configurations) and abstraction (e.g., requirements and specifications)
- For infrastructure – Foster consistent policies, standards, procedures, education, tools, incentives, and business practices that integrate reengineering into the systems engineering process

Technical reasons to reengineer existing software include mission requirements, major enhancements, modifications for new applications, rehosting to a new computer, translation to a new language, etc., which eventually come down to an economics decision. This Handbook provides the required DoD guidance for conducting technical, economic, and management assessments to determine when software reengineering techniques are cost effective.

1.4 Software Reengineering Assessment (SRA) Process

The factors leading to the decision to reengineer are complex. The goal of this handbook is to present those factors in a straightforward manner. Decisions regarding reengineering are critical since they involve the allocation of an organization's resources: *money*, *time*, and *personnel*.

Figure 1-1 presents an overview of the SRA Process. Subsequent chapters discuss the individual parts of the process in greater detail. There are multiple entry and exit points in the process. Candidate software (e.g., computer programs) enters the process based on a number of factors listed in Section 4 including those candidates perceived as being a problem. Candidates specifically directed to be reengineered will not need to be screened. Similarly, candidates that were previously identified through some other process for reengineering can enter the SRA process at the beginning of the economic process bypassing the technical process. However, the user may want to subject such candidates to the entire process to determine if a different strategy is more appropriate and/or to validate the process itself.

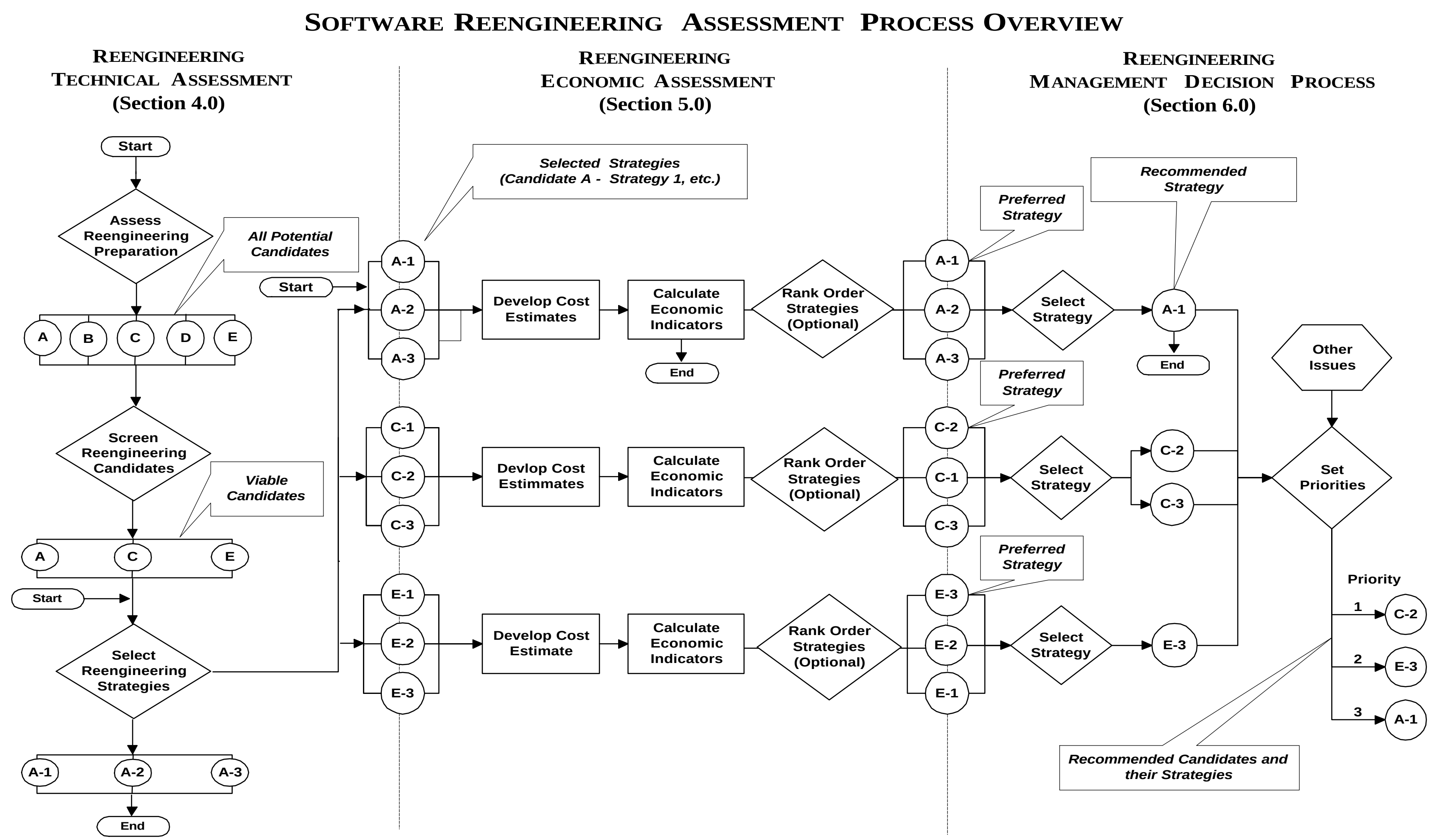


Figure 1-1. Software Reengineering Assessment (SRA) Process Overview

As previously described, the SRA process consists of three distinct components: *technical*, *economic*, and *management*.

A. Reengineering Technical Assessment (RTA) Process

1. Assess the organization's level of preparation to reengineer (Section 4.2).

2. Identify candidate software (Section 4.3).

3. Reduce the list of candidates to determine their viability for reengineering (Section 4.4).

4. Complete the RTA Questions, Table 4-2, for each candidate. Identify the reengineering strategies based on the scores from the questionnaire (Section 4.5).

5. Consider other strategies (Section 4.6).

6. If evaluating multiple candidates, enter the RTA Questionnaire results into the Detailed Assessment Results (DAR) worksheet in Appendix C, and answer the Maintenance Environment Question Set (Table 4.1).

7. Document the technical assessment results (Section 4.13).

B. Reengineering Economic Assessment (REA) Process

8. Establish ground rules and assumptions (Section 5.4).

9. Establish cost element structure (Section 5.5).

10. Determine estimating methods and collect data (Section 5.6).

11. Develop cost estimates (Section 5.7).

11.1 Perform cost sensitivity analysis (optional, Appendix D.1).

11.2 Perform cost risk analysis (optional, Appendix D.2).

12. Perform present value analysis and calculate economic indicators (Section 5.8)

13. Select preferred strategies (optional Section 5.9).

C. Reengineering Management Decision (RMD) Process

14. Prepare management report (Section 6.2).

15. Select the reengineering projects, and establish project priorities (Section 6.3).

16. Implement and document the management decision results (Section 6.4).

1.5 Content

Sections 1 through 6 contain the technical, economic, and management decision sections and related Appendices A through D. Appendices E through J, contains information, guidance, and examples for using specific models to estimate reengineering efforts. The examples were based on a case study which left much room for individual interpretation. Consequently, each model developer assumed slightly different attributes of the case study, which reengineering strategies were more beneficial/economical, and which approach should be taken for a particular strategy.

The results of the various models should not be compared with each other because the assumptions differed and the models are based on different project databases. The purpose of providing these appendices is to show that the models do support estimating reengineering efforts. In some cases, the model developers provided significant insight and innovation based on their experiences with actual reengineering projects to date.

2. Applicable Documents

2.1 Government Documents

2.1.1 Specifications, Standards, and Handbooks

This handbook references the following Government specifications, standards, or other handbooks:

a. *MIL-STD-881B, Work Breakdown Structures for Defense Materiel Items,* 25 March 1993.

b. *MIL-HDBK-171, Work Breakdown Structure for Software Element*, July 1995.

c. *MIL-HDBK-347, Mission Critical Computer Resources Software Support*.

d. *DoD-STD-2167A*, *Defense System Software Development*.

e. *MIL-STD-498*.

f. *MIL-STD-499*.

2.1.2 Other Government Documents and Publications

The following Government documents and publications are also referenced in this handbook:

a. *DoD 7041.3*, 18 October 1972, and *AFR 173-15*, 4 March 1988, "Economic Analysis and Program Evaluation for Resource Management."

b. *Department of the Army, Economic Analysis Manual*, U.S. Army Cost and Economics Analysis Center, Washington, D.C., August 1992.

c. *AFSC Cost Estimating Handbook*, Air Force System Command, Andrews AFB, D.C., Undated.

d. *OMB Circular A-94*.

e. *PDSS Cost Management and Process Control: Work Breakdown Structure, Cost Structure, and Data Collection, JLC/JPCG-CRM Draft Report*, 31 March 1994, Comptek Federal Systems, Inc.

f. *Proceedings of the First Software Reengineering Workshop, JLC-JPCG-CRM*, Santa Barbara I, September 1992, Joint Logistics Commanders.

g. Sittenauer, Chris, and Michael Olsem, "Back to the Future Through Reengineering: The Santa Barbara I Reengineering Workshop," *CrossTalk*, November 1992.

h. *Reengineering Technology Report*, Volumes 1 and 2, Software Technology Support Center (STSC), August 1993.

i. *Software Estimation Technology Report*, Software Technology Support Center, (STSC), March 1993.

There are numerous other DoD documents and federal documents which may provide additional information to help facilitate application of this handbook:

1. *DoD Enterprise Model*
2. *Technical Reference Model*
3. *Technical Architectures Framework for Information Management*
4. *CIM Software Systems Reengineering Process Model*
5. *DoD 8000.1 B Defense Information Management (IM) Program*
6. *DoD 8020.1-M B Functional Process Improvement (Functional Management Process for Implementing the Information Management Program of the Department of Defense)*
7. *DoD 8320 B DoD Data Administration.*
8. *A Plan for Corporate Information Management for the Department of Defense*
9. *CIM Business Process Improvement Program Implementation Pla*n
10. *Information Systems Criteria for Applying Software Reengineering*
11. *Functional Economic Analysis (FEA)* B Brochure
12. *Functional Economic Analysis Guidebook (CIM)*
13. *Process Improvement Methodology for DoD Functional Managers*
14. *User's Manual, Function Economic Analysis Model* (2.2.a).

2.2 Non-Government Documents and Publications

The following non-Government documents and publications are referenced:

a. Chikofsky, Elliot J., and James H. Cross II, "Reverse Engineering & Design Recovery: a Taxonomy," *IEEE Software*, pp. 13-17.

b. Boehm, Barry, *Software Engineering Economics*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

c. *PRICE SJ Reference Manual*, Third Edition, Martin Marietta PRICE Systems, Inc. Cherry Hill, NJ, October 1993.

d. *SEER-SEM User's Manual*, Galorath Associates, Inc., Los Angeles, CA, March 1996.

e. *Measures for Excellence: Reliable Software, on Time, within Budget*, Lawrence H. Putnam and Ware Myers, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1992.

f. *SLIM User's Manual*, Quality Software Measurement, Inc.

g. Londeix, Bernard, *Cost Estimation for Software Development*, Addison Wesley, 1987.

h. Wood, Michael, "A Reengineering Economics Model," *CrossTalk*, June/July 1992.

i. McCabe, et. al., "Structured Real-Time Analysis and Design," *IEEE COMPAC 85*, October 1985.

j. McCabe T., "A Complexity Measure," *IEEE Transactions on Software Engineering*, SE-2, 308-320, 1976.

k. *SoftCost-Ada User's Manual*, *Reference Manual*, and *Installation and Tutorial Manual*, Resource Calculations, Inc.

l. *CHECKPOINT User's Guide*, Software Productivity Research, Inc.

2.3 Reengineering Preparation and Project Planning Models

Several DoD organizations and private enterprises have defined reengineering preparation and project planning models. Two DoD agencies that have defined such models include:

- USAF Software Technology Support Center, OO-ALC/TISEC, 7278 Fourth Street, Hill AFB, Utah 84056-5205. Phone number (801) 777-8057 or DSN 458-8057.

- DISA/CIM, 701 South Courthouse Rd., Arlington, VA 22204-2199. Phone number (703) 285-6589 or DSN 356-6589.

3. Acronyms and Definitions

Analogy Estimating: An estimating method based on comparison of the software to be estimated with software that has been developed for known costs.

Assumptions: Statements or hypotheses made concerning uncertain factors and data that are required to complete the effort. They are not known facts.

Benefit: Results expected in return for costs and inputs incurred or used. A positive output of an alternative. Quantifiable benefits include cost savings, cost avoidance, and productivity improvement (personnel reductions). Non-quantifiable benefits include improved morale, quality, security, compatibility, and readiness *(USA Economics Analysis Manual)*.

Benefit Investment Ratio (BIR): The ratio of the present value of the dollar quantifiable benefits (savings and cost avoidances) divided by the present value of the investment cost (development, production, military construction, and fielding) of the alternative. It does not include benefits associated with sunk costs. *(USA Economics Analysis Manual).*

Break-even Point (BP): The point in time when the benefit in current dollars equals the investment in current dollars. It does not include sunk costs. Also, the point where the total cost of an alternative in current dollars equals the total cost of another alternative in current dollars. (*USA Economics Analysis Manual*).

Computer Software Component (CSC): A distinct part of a CSCI. CSCs may be further decomposed into other CSCs and CSUs *(DoD-STD-2167A)*.

Computer Software Configuration Item (CSCI): A configuration item for computer software *(DoD-STD-2767A).*

Computer Software Unit (CSU): An element specified in the design of a CSC that is separately testable *(DoD-STD-2167A)*.

Configuration Item (CI): An aggregation of hardware or software that satisfies an end use function and is designated by the government for separate configuration management *(MIL-STD-973).*

Constant-Dollars: All prior year, current, and future costs that reflect the level of prices of a base year. Constant dollars have the effects of inflation removed. *(USA Economic Analysis Manual)*. Constant dollars are normalized to the purchasing power of a dollar in a base year. Use of constant dollars allows consistent comparisons of costs between different years because of compensation for the effects of inflation and deflation.

Cost Avoidance: All reductions in future resource requirements, not in an approved POM or MDEP, because investment in some needed program/project will not have to be made *(USA Economics Analysis Manual)* (See Cost Savings).

Cost Element Structure (CES): The software product-oriented tree structure to which all life cycle cost accounts may be assigned; similar to a work breakdown structure (WBS).

Cost Savings: A cost reduction that is made in a specific POM or MDEP from implementing a specific alternative that does not degrade current capability, instead of the present system. Savings are a quantifiable benefit. (*USA Economics Analysis Manual*). USA Economics Analysis policy defines cost savings as a benefit occurring only during a POM or MDEP period, and cost avoidance as a benefit occurring both during and following a POM or MDEP period but not part of either. Cost avoidance and cost savings are mutually exclusive.

Current Dollars: Dollars that reflect the purchasing power of the dollar in the year the cost or savings is to be realized or incurred. That is, current dollars reflect the effects of inflation. Prior year costs stated in current dollars reflect actual present-day costs. Future costs or savings stated in current year dollars are the projected values which will be paid out or saved in future years. *(USA Economics Analysis Manual)* Prior, current, and future dollars that have been adjusted to reflect the reduced purchasing power of dollars received in the future. This reflects the fact that dollars received or spent in the future are worth less than dollars today; therefore, more dollars will be needed in the future to purchase the same item. Current Dollars are calculated from specific Constant Dollars using weighted inflation rates.

Cyclomatic Complexity: A measurement of the number of linearly independent paths through a program: the number of control verbs plus one. (See "A Complexity Measure", *IEEE Transactions on Software Engineering*, SE-2, 308-320, 1976.

Data Reengineering: Transformation of data from one format to another format (e.g., going from flat-file to relational database format) (Santa Barbara I).

Data Standardization: The "process of reviewing and documenting the names, meanings, and characteristics of data elements so that all users of the data have a common, shared understanding of it. Data standardization is a critical part of the DoD Data Administration Program, managed under *DoD Directive 8320.1*. Data administration is the function that manages the definition and organization of the Department's data" [Memorandum data October 13, 1993 subj: "Accelerated Implementation of Migration Systems, Data Standards, and Process Improvement," page 3].

Discounting: A technique for converting various annual cash flows over a period of time to equivalent amounts at a common point in time, considering the time value of money, in order to facilitate comparison (*USA Economic Analysis Manual*).

Discounted Dollars: Future costs that have been discounted to reflect the time value of money.

Discount Rate: The interest rate used to discount or calculate future costs and benefits so as to arrive at their present values *(DoD 7041.3)*. This term is also known as the opportunity cost of capital investment (*USA Economic Analysis Manual)*.

Economic Assessment: A process for evaluating the relative cost differences between candidate strategies for reengineering software.

Economic Indicators: Metrics that can be used to appropriately compare various alternative strategies being considered. These indicators include Present Value of Total Cost PV(TC), Present Value of Total Benefits PV(TB), Net Present Value (NPV), Benefit Investment Ratio (BIR), Break-even Point (BP), and Rate of Return (ROR). NPV is preferred.

Essential Complexity: A measurement of the level of "structuredness" of a program. (See "Structured Real-Time Analysis and Design" by McCabe et. al., *IEEE COMPSAC-85*, October 1985.)

Existing Software: Existing source code, assembly code, documentation, or code implemented in firmware being evaluated for potential reengineering.

Existing System: The existing hardware, software, and/or firmware that is being evaluated for potential reengineering.

Forward Engineering: The set of engineering activities that consume the products and artifacts derived from existing software and new requirements to produce a new target system (Santa Barbara I).

> Note: Notice the difference between software engineering and forward engineering. An issue within the software reengineering community is whether the term "forward engineering" is needed since it implies the normal development life cycle sequence of events. If this were true, then forward engineering and software engineering can be considered identical terms. But forward engineering can be defined as using the output of reverse engineering. This implies some reengineering must have occurred prior to the forward engineering activity. For example, the most common forward engineering activity involves the generation of source code from design information which was captured by a previous reverse engineering activity.

Function Points: The unit of measure used in Function Point Analysis (FPA). It is a technique for measuring the "size" of a software project or system. It is based on the functionality of the software expressed in terms of five components: (1) internal logical files, (2) external interface files, (3) external inputs, (4) external outputs, (5) external inquiries. Each component is rated according to a qualifying criteria provided in the *International Function Point Users Group (IFPUG) Counting Practices Manual* (*STSC Software Estimation Technology Report*, "Appendix G," March 1993).

Ground Rule: A basic rule or fact that cannot be altered and which constrains an analyst from considering an alternative course of action.

Initial Operating Capability (IOC): The date at which operational acceptance testing is complete and when support begins. The first capability attainment to imply effectively a weapon system, an item of equipment, or system of approved characteristics, and which is manned or operated by a trained, equipped, and supported military unit (Defense Systems Management College).

Investment Cost: Includes Research and Development (R&D) and Production and Deployment costs of the system (*USA Economic Analysis Manual*). Cost elements include software development, site preparation, training, development tools, and hardware. Investment does not include the Operations and Support (O&S) cost to maintain either the existing system or the reengineered system.

JLC-JGSE: Joint Logistics Commanders, Joint Group on Systems Engineering.

Legacy System: One of a set of "other AISs...that duplicate the support services provided by the migration system." [Legacy systems] are terminated, so that all future AIS development and modernization can be applied to the migration system (Deputy Secretary of Defense memorandum dated October 13, 1993; subj: "Accelerated Implementation of Migration Systems, Data Standards, and Process Improvement," page 4).

> Note: The term "legacy software" arises from the fact that these systems are frequently unstructured, undocumented, and are the legacy of earlier, less mature days of software development.

Level of Effort (LOE): An estimating method based on the number of labor hours and a fixed labor rate.

Life Cycle Cost Estimate (LCCE): Includes all costs incurred during the total life of a project, from initiation through termination. The LCCE includes the costs for Research and Development, Production and Deployment, and Operations and Support (*USA Economic Analysis Manual*).

Lines of Code (LOC): A measure of the size of computer software; a significant cost driver for development and maintenance; see also SLOC.

> Note: LOC as a size metric has been criticized as being ambiguous and even meaningless. While it is widely used, it may not be wise to rely on it as a sole measure of size. If a decision is made to use LOC, care should be taken to ensure it is consistently applied across projects and that differences in languages are taken into account.

Maintenance/Support: The process of modifying the existing operational software while leaving its primary functions intact. Software maintenance can be classified into two main categories: (1)

Update, which results in a changed functional specification for the software product, and (2) *Repair*, which leaves the functional specifications intact. In turn, *Repair* can be classified into three main subcategories: (2a) *Corrective maintenance* (of processing, performance, or implementation failures), (2b) *Adaptive maintenance* (to changes in the processing or data environment), and (2c) *Perfective maintenance* (for enhancing performance or maintainability) (Boehm 81).

Management Decision Package (MDEP): A structured life cycle process that represents the most current funding position developed through the Planning, Programming, Budgeting, and Execution System (PPBES). A separate MDEP will normally be created for each AIS. Each MDEP covers a nine-year period (*USA Economic Analysis Manual*).

Management Decision Process: A process for assimilating the results of the Technical and Economic Assessments (typically by technical questionnaire scores or economic indicators) to select the recommended strategy for each program and for prioritizing the list of programs.

Migration System: "An existing automated information system (AIS), or a planned and approved AIS, that has been officially designed as the single AIS to support standard processes for a function." "A migration system is designated (or selected) by the OSD Principal Staff Assistant(s) and their Defense Component counterparts whose function(s) the system supports, with the coordination of the DoD Senior Information Managements Official." From Deputy Secretary of Defense (memorandum dated October 13, 1993 subj: "Accelerated Implementation of Migration Systems, Data Standards, and Process Improvement," page 4).

Net Present Value: Traditionally, the difference between the present value of the dollar quantifiable benefits and the present value of the cost. (*USA Economic Analysis Manual*). It indicates the "net benefit or cost" of an alternative compared to another alternative.

New Development: The engineering process of developing new software by starting over at the requirements analysis phase without reusing existing requirements specifications, design, or code. An extreme case of redevelopment rarely occurring.

Non-Developmental Item (NDI): Items not requiring development *(DoDI 5000.2)*.

Non-Quantifiable Benefit: A benefit that is not adequately reflected using mathematical or numeric representation, such as better quality of services, and which is typically better expressed in narrative form (*USA Economic Analysis Manual*).

Parametric Cost Estimating Model: An aggregate of cost estimating relationships (CERs) which calculates predicted development and maintenance costs based on technical and schedule cost drivers.

Preferred Strategy: Ideally, a candidate strategy which has the highest Net Present Value (NPV) and/or the highest Benefit Investment Ratio (BIR).

Present Value Analysis: An analysis of competing alternatives using their present value.

Present Value: Dollars that have had their annual cash flow over time converted to equivalent amounts at a common point in time in order to account for the time value of money. The discount rate is prescribed by OMB. The computation begins with constant dollars which are multiplied by the discount rate (*USA Economics Analysis Manual*).

Program: Computer software (source code, assembly code, or code implemented as firmware).

Program Objective Memorandum (POM): See Management Decision Package, but for a five-year period.

Quantifiable Benefit: A benefit that can be assigned a numeric value such as dollars, physical count of tangible items, or percentage change (*USA Economic Analysis Manual*).

Ranking (or Rank Ordering): The relative ordering of candidate strategies based on an economic or technical metric (e.g., rank order #1 would be assigned to the most economically beneficial strategy).

Rate of Return (ROR): The interest rate at which the present value of the savings (benefit) equals the present value of the investment cost through the remaining life cycle of the alternative being evaluated (*USA Economic Analysis Manual*).

Recommended Strategy: The candidate strategy which is selected in the Management Decision Process.

Redevelopment: The engineering process of developing a new system by starting over at the preliminary design phase and reusing the existing requirements specifications.

Redocumentation: The process of analyzing the system to produce support documentation in various forms including users manuals and reformatting the system's source code listings (Santa Barbara I).

> Note: Sometimes, the output of reverse engineering is thought to be the same as redocumentation. After all, when reverse engineering captures the design information from the existing source code, the resulting display can take the form of data flow diagrams, control flow charts, etc. The difference between redocumentation and reverse engineering is that the redocumentation usually generates system documentation according to a standard. For example, there are redocumentation tools that create documentation for *MIL-STD-2167A*, the standard for documenting DoD software development projects.

Reengineering: The examination and alteration of an existing subject system to reconstitute it in a new form. The process encompasses a combination of subprocesses (strategies) such as reverse engineering, translation, restructuring, data reengineering, redocumentation, forward engineering, and retargeting, (Santa Barbara I).
Remaining Life (years): The required lifetime, starting when the reengineering effort begins. The sum of the reengineering time plus the operational lifetime of the reengineered software.

Restructuring: The engineering process of transforming the existing system from one representation form to another at the same relative abstraction level, while preserving the subject system's external functional behavior (Santa Barbara I).

> Note: In contrast to reverse engineering (which takes in source code and extracts design information which is a higher level of abstraction than the code), restructuring code leaves the system at the same abstraction level (within the life cycle) of "code," but radically rearranges the source code to fit a new paradigm such as structured analysis or object-oriented analysis.
>
> Restructuring is commonly used to mean taking unstructured code and making it structured. Architecture level changes need their own term and include changes to interfaces of modules, reduction of global data, remodularization of functionality, etc.

Retargeting: The engineering process of transforming, rehosting, or porting the existing system in a new configuration (Santa Barbara I).

> Note: The new configuration could be a new hardware platform, a new operating system, or a Computer Automated Support Environment (CASE) platform. In some cases utilization of CASE tools will not require a change of platform and the term retargeting would be misleading in cases where only a software change in the support environment is performed.

Retirement: Discontinuing use of a system, without revision (Santa Barbara I).

Reverse Engineering: The engineering process of understanding, analyzing, and abstracting the system to a new form at a higher abstraction level (Santa Barbara I).

> Note: This higher abstraction level is understood within the context of the software system's life cycle. For example, the classic waterfall development life cycle calls for requirements/specifications followed by design, code, test, implement, and maintain. Thus, if we start with the code, reverse engineering will extract design information which is at a higher abstraction level in the life cycle.

Source Lines of Code (SLOC*):* A metric for software size. Generally defined as executable source code statements. Specific definitions vary among parametric software estimating models, and depend on the implementing language (See Lines of Code).

Source Code Translation: Transformation of source code from one language to another or from one version of a language to another version of the same language (e.g., going from COBOL-74 to COBOL-85 or from CMS-2 to Ada) (Santa Barbara I).

Status Quo: Continued maintenance (support) of the existing software. In the Economic Assessment, this is always Strategy #1 (Santa Barbara I).

"Sunk" Costs: Past expenditures or irrevocably committed costs which are unavoidable and, therefore, should not be considered in the decision process (*USA Economic Analysis Manual*).

Systems Engineering: The top level process of engineering a system (hardware and software) to meet overall requirements.

Technically Acceptable Strategy: A strategy identified by the Technical Assessment Process as a candidate strategy for addressing the reengineering requirement. It must be a strategy that meets all related technical requirements.

Technical Assessment: A process for screening and evaluating attributes of an existing computer program to identify candidate strategies. These candidate strategies then will be economically assessed.

Technical Indicators: Metrics, derived during the technical assessment, that identify which candidate strategies have comparatively the greatest technical need to reengineer.

Then-Year Dollars: See Current Dollars.

Total Benefit (TB): The difference between the total O&S cost for Strategy 1 and the total O&S cost for Strategy N (including the O&S cost for Strategy 1 during reengineering).

Total Cost (TC): The sum of investment (CES 1.0) and Operations & Support (CES 2.0) cost for the remaining life of the software, excluding sunk costs. For reengineered software, Total Cost includes the O&S cost to maintain the existing software during reengineering. The analysis must include the continued use of the existing system until replaced by the alternative as part of the cost of the alternative (*USA Economics Analysis Manual*, paragraph 2-10).

Uniform Annual Cost: A constant dollar amount which, if paid annually throughout the economic life of a proposed alternative, would yield a total discounted cost equal to the actual present value of the alternative. It is determined by dividing the total discounted alternative cost by the sum of the discount factors for the years which an alternative yields benefits. It is used to compare alternatives with different economic lives (*USA Economic Analysis Manual*).

4. Reengineering Technical Assessment

4.1 Introduction

The Reengineering Technical Assessment (RTA) is the first part of the Software Reengineering Assessment Process. The primary purpose of this section is to match existing software components (otherwise known as "legacy" software or "candidate" software systems, programs, subroutines, etc.) with appropriate software reengineering or maintenance strategies. RTA is primarily focused on the technical aspects of legacy software systems. Therefore, it is recommended that this section be completed by the organization's technical personnel. RTA results are then passed on to Section 6 of this handbook to aid in management's reengineering decisions.

This handbook currently recognizes eight basic software maintenance strategies. Six of these strategies are considered reengineering strategies and two are classic maintenance strategies.

The six reengineering strategies are as follows:

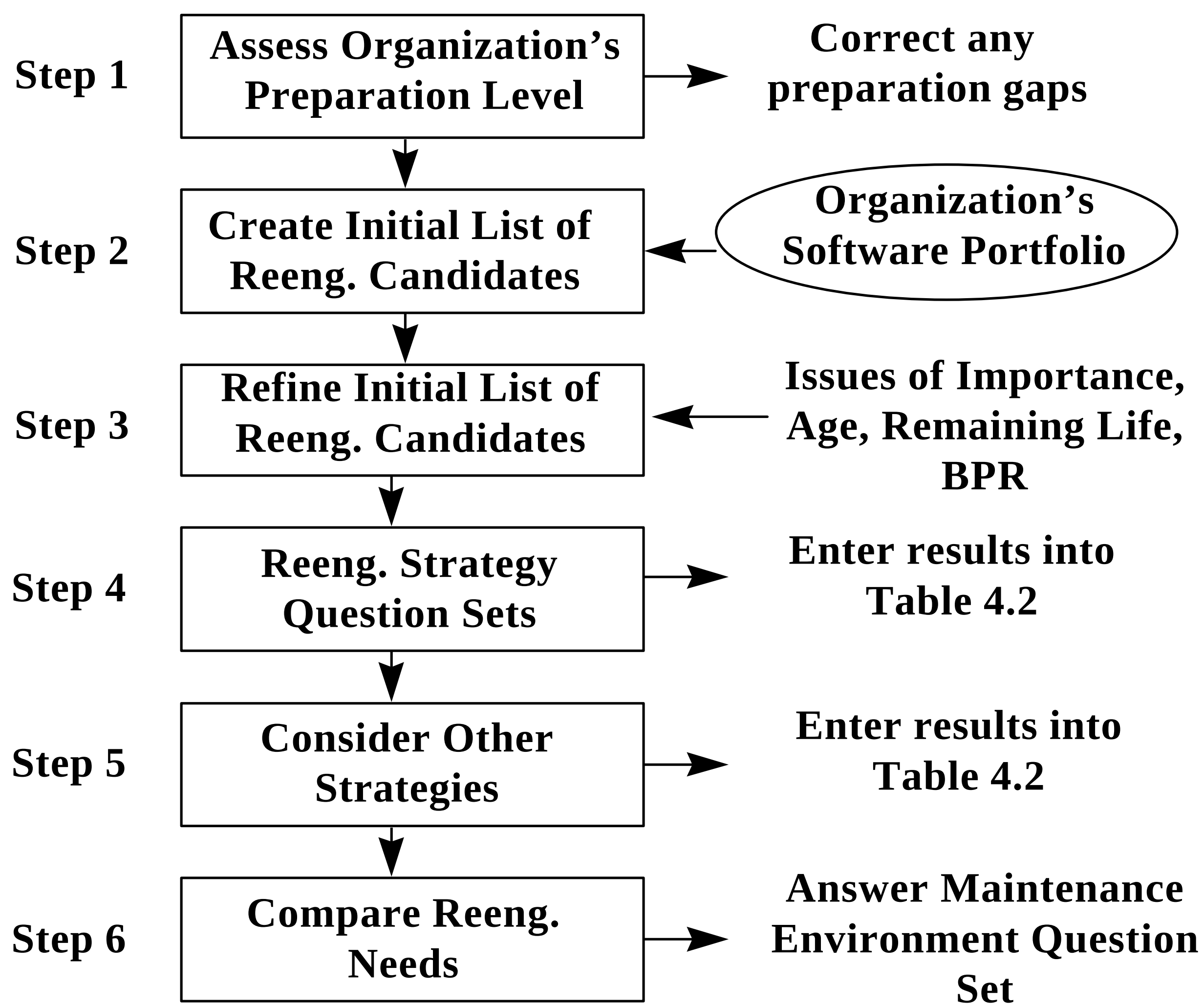
- Redocument
- Reverse Engineer
- Translate Source Code
- Data Reengineer
- Restructure
- Retarget

The two classic maintenance strategies are as follows:

- Redevelopment
- Status Quo

The RTA will provide a quick and easy method for determining which of the above eight software maintenance strategies is best suited to a given candidate software component.

Figure 4-1
Reengineering Technical Assessment
Flow Chart



4.2 Step One: Assess the Organization's Level of Preparation

By studying successful and unsuccessful software reengineering projects, this handbook has determined the minimum criteria an organization should meet to help ensure a successful reengineering project.

The question set entitled "Reengineering Preparation" beginning on page 4-14, has been created to help assess an organization's level of preparedness to reengineer. By adding up the answers of each question and dividing by the number of questions answered, determine the average response per question (use the RTA Summary Worksheet Table 4.2).

If the average is below 2.00, this handbook strongly recommends that an organization correct its level of preparation using this question set as a rough guide for what needs to be addressed. If the average is between 2.00 and 3.00, circle "yes" next to "Preparation" on RTA Summary Worksheet 4.2 for the candidate software. Otherwise circle "no" next to "Preparation" on the RTA for the candidate software. ***Do this before using any additional part of this handbook!*** Without a high level of preparation, the reengineered software will, at best, require another round of reengineering in a few years and, at worst, the reengineering project will fail.

4.3 Step Two: Identify Software Reengineering Candidates

From the portfolio of all the software systems within the organization[1], identify and list those systems, programs, or components which are perceived as being a problem, or having significant improvement potential. Look at the organization's base of existing software with the following issues in mind:

- Age - is the software very old?
- Complexity - is the software too complex (using complexity measurement tools)?
- Language - is the language out of date, not portable, or cumbersome to maintain?
- Reliability - does the software fail periodically?
- Level of maintenance required (in dollars or man-hours) - is the software difficult or expensive to maintain?
- State of documentation - is the documentation poor and/or inadequate?
- Impact if software should fail - will the customer be severely impacted by the software's failure?
- Degree of coupling between hardware and software - is the software's coupling with the hardware blocking future enhancements?

[1]This assumes there exists a current portfolio of all software systems in use. If not, create one prior to this step.

- Personnel knowledge, experience, and turnover - is there currently (or in the near future) a lack of experienced maintenance programmers for this software?
- Technological constraints - is the software or hardware blocking future enhancements?
- Change of platform...past or future - has a change of platform occurred for the software or will a change soon occur?

4.4 Step Three: Reduce the List of Software from Step Two

The list of candidate reengineering software from step two may still be long. *Remove a software candidate from the above list if any of the following criteria apply:*

- The remaining life of the candidate software is projected to be less than 3 years.

  Reengineering usually requires significant start-up resources (personnel, training, software, and hardware). These reengineering costs can be amortized over the remaining life of the software since software reengineering will reduce annual software maintenance costs. Typically, at least three years are required before an organization will see a return on its reengineering investment. Therefore, if remaining life is greater than three years, this handbook suggests circling "yes" next to "Remaining Life" on RTA Summary Worksheet for the candidate software. If not, circle "no" next to "Remaining Life" on RTA Summary Worksheet for the candidate software.

- The candidate software isn't important enough to reengineer.

  The candidate software should be evaluated against the question set "Software Importance". The average of answers for this question set should fall within the range of 2.00 to 3.00 to meet this importance criteria. If so, circle "yes" next to "Importance" on RTA Summary Worksheet 4.2 for the candidate software. If the average is less than 2.00, drop the software from the list of candidate reengineering software and circle "no" next to "Importance" on RTA Summary Worksheet 4.2 for the candidate software.

- The candidate software was first developed less than 5 years ago.

  This assumes modern programming techniques (such as structured programming or object-oriented design) are now being used and have been used over the past five years. Newer software has the additional advantage of access to the original development staff who understand the software for easier maintenance. Reengineering shows the best return when used on candidate software that is difficult to maintain. Therefore, if the age of the software is greater than five years, this handbook suggests circling "yes" next to "Age" on RTA Summary Worksheet 4.2 for the candidate software. If not, circle "no" next to "Age" on RTA Summary Worksheet 4.2 for the candidate software.

- The candidate software directly supports a business process currently being reengineered.

  If the organization is undergoing business process reengineering (BPR) and the candidate software directly supports a process being reengineered, then don't reengineer that software until it is determined what software changes are needed. BPR will fundamentally change the way an organization functions. Software reengineering supports BPR. Thus, any software that supports a business process undergoing BPR will also change fundamentally. The final process must be fully defined before changing the software that supports that process.

Candidate software, rejected for reengineering due to one or more of these criteria, may be retained on the list of reengineering candidates if there are extenuating circumstances (management demands, customer pressure, funding source restrictions, etc.). Additional criteria, unique to each organization, may also be applied to the list from step two in order to reduce the number of candidates for reengineering.

4.5 Step Four: Complete the Reengineering Technical Assessment RTA Questions

For each remaining candidate software, answer the question sets corresponding to the following reengineering strategies (create copies of the questions sets for each candidate software component to be evaluated for reengineering):

- Redocument - page 4-19
- Translate Source Code - page 4-20
- Data Reengineer - page 4-22
- Retarget - page 4-24
- Restructure - page 4-26

Consult with personnel familiar with the candidate software. Group consensus helps to minimize potential error or bias. After answering as many of the questions as possible, compute the total for each question set and divide by the number of questions answered (use the accompanying RTA Summary Worksheet). This average response for each question set will determine whether that reengineering strategy is a good match for the candidate software being considered. A few skipped questions, which cannot be answered, will not significantly affect the reengineering assessment score since other questions are designed to elicit similar information from a different perspective.

- If the average falls within the range of 2.40 to 3.00, then that reengineering strategy is a very good match for the candidate software. Circle "yes" under "Indication of Strategy" for that strategy.

- If the average falls within the range of 1.70 to 2.39, then some benefit is indicated by applying that reengineering strategy but the match is not solid. There may be other strategies better suited to this candidate software. Circle "maybe" under "Indication of Strategy" for that strategy.

- If the average falls within the range of 1.00 to 1.69, then the reengineering strategy is a poor match for the candidate software. Circle "no" under "Indication of Strategy" for that strategy.

4.6 Step Five: Consider Other Strategies

Note that the RTA Summary Worksheet includes three additional strategies. This section will explain when these strategies should be considered.

4.6.1 Reverse Engineer

Reverse engineering is the process of extracting design information from source code, storing the extracted information in a repository, and graphically presenting this information (such as a control flow diagram, data flow diagram, etc.). Reverse engineering is a larger, more complex, and usually more expensive strategy than any of the other five reengineering strategies alone. Many organizations that decide to reverse engineer use the resulting design information repository as a key element to their long term maintenance process. Such organizations maintain their software by modifying the repository information instead of the source code. The executive Summary Worksheet source code can then be generated from the altered design information.

Since reverse engineering is a larger, more complex, and usually more expensive strategy, it is normally used only when one of the following conditions are met:

- Multiple reengineering strategies are indicated.

  Reverse engineering can be used as a common, interim step when two or more reengineering strategies are indicated from RTA Summary Worksheet. For example, if restructuring and source code translation are indicated, reverse engineering could facilitate the other two strategies in the following way:

  First, capture the design information using reverse engineering. Understand how the existing software functions. Clean up the candidate source code at the design level, eliminating any "dead" code (non-executable). Second, generate the new source code language from the improved design information. Restructure the generated source code using restructuring tools created by vendors for the new source code language. Third, capture the restructured design information back to a design repository.

This process allows the maintenance programming staff to always understand how the software works even after it's translated to a new language and the control flow is radically altered by restructuring.

- The control flow of the candidate software is too complex to understand

  The documentation is missing or wrong and years of haphazard maintenance has created software that cannot be modified without creating unwanted side-effects. Studies show that software understanding is 70% of most maintenance activities. As described in the previous bullet, reverse engineering allows the maintenance staff to understand the control flow and eliminate unused code.

- Reuse is a key objective of the organization

  Source code repositories have always been key to an organization's reuse strategy. However, the reusable code fragments stored in such repositories are usually not fully trusted by the development staff. They do not know exactly how each reusable code fragment functions, and how it will interface with the rest of their new software. Although there's documentation, it's questionable whether it's accurate, complete, and current.

  On the other hand, a design repository's reusable *design* fragments will inherently show the developers what's going on inside each fragment. Such fragments can be combined and altered to create a larger software design. The resulting design can then be translated into executable source code using forward engineering tools.

In conclusion, the reverse engineering strategy is indicated if any of the preceding conditions are met. If so, circle "yes" next to "Reverse Engineer" on RTA Summary Worksheet for the candidate software. Otherwise, circle "no" next to "Reverse Engineer" on RTA Summary Worksheet for the candidate software.

4.6.2 Redevelopment

Redevelopment is the traditional, non-reengineering solution for really bad source code. The Redevelopment option is used when the candidate software (including its design) is not worth salvaging and should be developed from scratch. If three or more reengineering strategies are indicated using the associated question sets, and the projected remaining life of the candidate software exceeds five years, then seriously consider redevelopment for the candidate software. Three or more reengineering strategies may prove as expensive as redevelopment and may indicate that the software is not worth salvaging by reengineering techniques.

There are certain advantages to redevelopment. "Getting it right this time" using modern programming practices certainly leads the list. Of course, this assumes the organization's current development and maintenance practices are now well-defined and enforced. Maintenance costs will generally be significantly lower once the redevelopment is completed. Lower even than if this software had been reengineered instead of redeveloped.

The disadvantages reflect a significantly higher cost to redevelop as compared to reengineering. That is why five years is generally considered the minimum number of years to recoup redevelopment investment due to much lower maintenance costs (similar to the reason why "newer" programs of less than five years since development are generally not considered as candidates for reengineering).

Thus, if both conditions are met (three or more reengineering strategies are indicated *and* the remaining life is more than five years), circle "yes" next to "Redevelop" on RTA Summary Worksheet for the candidate software. Otherwise, circle "no" next to "Redevelop" on RTA Summary Worksheet for the candidate software.

4.6.3 Status Quo

If the question sets determine that no reengineering strategy is needed, then leaving the software alone may be the best policy. If this is the case for the candidate software, circle "yes" next to "Status Quo" on the RTA Summary Worksheet. If any reengineering strategy is indicated, circle "no" next to "Status Quo".

4.7 Step Six : Comparing Reengineering Needs Across Different Software Components

Now that the reengineering needs of a given software component have been assessed, different software components (and their respective reengineering strategies) must be compared. For example, software component 'A' may indicate a need to restructure and retarget while software component 'B' may indicate a need to translate source code and data reengineer. How does component 'A' and its reengineering needs compare to component 'B' and its reengineering needs?

4.7.1 Management Decision Aid.

Enter the reengineering strategy question set results from RTA Summary Worksheet into the management decision worksheet found in Section 6 of this handbook. This technical assessment information will be combined with the cost data for each reengineering strategy (Section 5) and then both will be considered by management (Section 6).

4.7.2 Maintenance Environment

As an additional factor for comparing reengineering needs between software components, answer the Maintenance Environment Question Set on page 4-28. This set of questions assesses the current maintenance environment of the candidate software. Since reengineering generally results in software that is more easily maintained, this question set considers other factors that may raise (or lower) the urgency to reengineer a particular software candidate.

If the average of answers for this question set falls within the range of 2.00 to 3.00, circle "yes" next to "Maintenance Environment" on RTA Summary Worksheet for the candidate software. If the average is less than 2.00, circle "no" next to "Maintenance Environment" on RTA Summary Worksheet for the candidate software.

4.8 An Example Using the SRAH Reengineering Technical Assessment

The following is an example of applying the SRAH Reengineering Technical Assessment Process.

Step one: After answering only 24 of the Reengineering Preparation questions, you add up the 24 answers and discover a total of 59 points. Divided by 24, the average answer is 2.46. Since 2.46 is above the 2.00 threshold, you decide that your organization has done sufficient planning (i.e. is prepared) for a reengineering project. Thus, you circle "yes" next to Preparation in RTA Summary Worksheet. However, you notice a few questions in the Preparation Question Set in which you had to answer "1". Mark these for immediate attention and plan to correct them. Enter the numbers 24, 59, and 2.46 within their respective columns for Preparation in worksheet 4.2. These numbers will be the same for all candidate software components so you don't have to re-answer the Preparation Question Set.

Step two: Based on criteria listed in section 4.3, compile a list of 33 software components that warrant attention. Some of these software components are maintenance nightmares, others are very old systems that are inhibiting your organization from taking advantage of newer hardware platforms or CASE tools.

Step three: Of the 33 "problem" software candidates, you discover 3 were developed only 2 years ago. (Make a mental note to discover why these relatively new software components were so poorly developed). Another 6 are due to be discontinued within a few years due to customer disinterest or future funding constraints. While 5 more directly support the accounting system which will soon be overhauled due to an upcoming reorganization of that business sector. This leaves 19 candidate software components (33 minus 14 equals 19).

By applying the Importance Question Set, against each of the remaining 19 candidate software components, you discover that only 7 are important enough to spend critical organizational resources (personnel and funds) to warrant reengineering. Of these 7, you begin with candidate software component "A". For software component "A", the sum of your 5 answers for the Software Importance Question Set equals 13 for an average of 2.6. So you circle "yes" next to Importance in RTA Summary Worksheet and write a brief description of software component "A" in the upper left corner of this worksheet. Enter the numbers of 5, 13, and 2.6 within their appropriate columns for Importance on RTA Summary Worksheet.

Having passed the Age and Remaining Life criteria, circle "yes" on RTA Summary Worksheet for these two items for software component "A".

Step four: For software component "A", answer the question sets corresponding to the five reengineering strategies of redocument, data reengineer, restructure, translate source code, and retarget.

Results:

- redocument:
  - 5 questions answered, sum of 14, average of 2.8
  - circle "yes"
- translate source code:
  - 6 questions answered, sum of 15, average of 2.5
  - circle "yes"
- data reengineer:
  - 9 questions answered, sum of 12, average of 1.33
  - circle "no"
- retarget:
  - 5 questions answered, sum of 9, average of 1.8
  - circle "maybe"
- restructure:
  - 6 questions answered, sum of 9, average of 1.5
  - circle "no"

Step five: You now need to consider whether reverse engineering, redevelopment, or status quo are valid strategies as well.

Reverse engineer:

Two reengineering strategies were indicated (redocument and translate source code) and another was possibly indicated (retarget). Since two "yes" indicate that reverse engineering should be considered, circle "yes" next to this strategy on the worksheet. Your strategy may be to reverse engineer the candidate software, thus capturing its control flow and other design information. This design information can now be used to document the candidate software. When translating source code, you need to take advantage of the power of the new language. To do this, you could manipulate the design information within the design repository then use a source code generator to create executable source code in the new language.

Redevelopment:

Two "yes" and one "maybe" indicate a borderline possibility of redevelopment. Decide whether to consider redevelopment when doing cost estimates in section 5 of this handbook.

Status Quo:

At least two reengineering strategies are indicated so circle "no".

You answered 7 of the Maintenance Environment questions for a total 12 points or an average of 1.71. Circle "no" next to "Maintenance Environment" on RTA Summary Worksheet since the average is less than 2.00.

Repeat this process, using copies of the worksheet, for the other six candidate software components. Transfer the results to the appropriate worksheets in Sections 5 and 6.

4.9 The Importance of a Pilot Project

As with any new technology, reengineering should be inserted cautiously within an organization. After choosing the prime software candidates for reengineering, choose one that is small enough and isolated enough (little or no impact on other software components and data files) to make a good pilot project. Such a pilot project enables an organization to test the newly acquired tools and newly defined techniques of reengineering. Lessons learned from pilot projects can then be applied to larger, more complex software reengineering projects.

4.10 Impact Analysis

Several reengineering strategies will substantially change existing source code. Since many software components do not stand alone, management must be advised of the impact of reengineering on dependent software components and data files.

For example, source code translation from COBOL to Ada will impact how the translated software interfaces with its data files. If those data files are IMS and the existing software uses embedded IMS macro calls, then the organization must consider how the newly translated software will now access those same data files. Such considerations may require data reengineering of the existing data files to a format accessible by the new software. IMS-like macro calls for the Ada compiler may exist. If the organization ends up data reengineering the existing data files, what impact will this have on other software systems that use these same data files?

Therefore, this handbook recommends listing and defining all data files and dependent software systems that interface with the reengineering candidate software. Carefully explain how each reengineering candidate accesses its data files and passes information to other software systems. In this way, management will understand the full scope of the proposed reengineering projects.

4.11 Additional Reengineering Considerations

Two reengineering strategies may have some difficulty if implemented alone. These strategies are restructuring and source code translation.

Restructuring will radically alter the internal control flow of the existing software. The maintenance staff, currently familiar with the control flow of the existing software, will have initial difficulty understanding the newly restructured software. Thus, an experienced maintenance programmer and one unfamiliar with the software will be on the same footing. Some organizations reassign such restructured code to a new set of maintenance programmers. This handbook recommends that some form of reverse engineering accompany restructuring of source code. A graphical representation of the control flow, before and after the restructuring, will facilitate software understanding and future maintenance requests.

Source code translators typically perform line-for-line translations. This approach does not take advantage of the advanced semantic constructs inherent in the new language. For example, when translating source code from COBOL to Ada, line-for-line translation will not take advantage of Ada's object-oriented constructs. This handbook recommends restructuring take place *after* the source code translation. Restructuring tools are written with specific languages in mind. Thus, most restructuring tools will help adapt the translated code to the new language. Consider reverse engineering as an interim step for source code translation. After reverse engineering, one could manipulate the graphical design representation to fit the new language's constructs, then generate the new source code from the design repository.

4.12 Software Analysis Tools

All reengineering strategies should include tools for analysis of candidate software prior to reengineering. Analysis tools can:

- Identify "dead" (unused) code
- Measure the candidate software's level of complexity
- Identify poor coding practices (such as uninitialized variables, etc.)
- Measure the number of function points
- Check for non-structured constructs

Cleaning up the candidate software prior to reengineering will facilitate the reengineering project by eliminating problem code before subjecting the software to the reengineering process. It makes no sense to translate, restructure, redocument, reverse engineer, or retarget bad source code embedded within the candidate software.

4.13 Reporting the Results of this RTA

All the work accomplished by the technical staff during this RTA should be reported to management using section 6 of this handbook. RTA Summary Worksheet will be used by Sections 5 and 6 of this handbook. Additionally, recommendations based on sections 4.9 through 4.12 should also be passed along to management. Write a short report containing these technical assessment findings as an addendum to the final report prepared for management in Section 6. This addendum should include the results of RTA Summary Worksheet and Sections 4.9-4.12 with a brief explanation for each finding.

TABLE 4-1 REENGINEERING TECHNICAL ASSESSMENT QUESTIONS

	REENGINEERING PREPARATION
	Organizational Needs 1. Have the purposes/goals of reengineering been defined for this reengineering project? 1 - No 2 - Yes but they're unclear to me or don't know them 3 - Yes 2. Does management support this reengineering effort? 1 - No 2 - Yes but with reservations or management perceives that reengineering will solve all their maintenance problems 3 - Yes 3. How important is this reengineering project to the organization (managers, users, etc.)? 1 - Not important 2 - Average importance 3 - Critical 4. Is there financial support for this reengineering effort? 1 - Expect little or no financial support 2 - Expect some funding but not sufficiently funded 3 - Expect project will be fully funded 5. Is the reengineering effort consistent with overall corporate strategy? 1 - Not aware of a corporate strategy 2 - Mostly 3 - Yes **Reengineering Team** 6. Has a reengineering team been chosen? (Picked from the users, maintenance personnel and management) 1 - No or not representative of all key groups 2 - Yes but not dedicated full time 3 - Yes

	New Maintenance Process Defined 7. Has a new maintenance process environment been clearly documented? 1 - No 2 - Yes but only in broad, general terms 3 - Yes 8. Has an organizational CMM software assessment been done? 1 - No 2 - Yes but unknown 3 - Yes 9. The organization's CMM level is: 1 - One 2 - Two 3 - Three or above 10. Has a tactical plan been defined and implemented to improve the CMM level of the maintenance organization? 1 - No tactical plan to improve maintenance 2 - Yes, the tactical plan has been defined 3 - Yes, the tactical plan is being implemented **Metrics** 11. Have development/maintenance software metrics been defined for the organization? 1 - No 2 - Yes but the metrics are not based on process improvement or related to corporate goals 3 - Yes 12. Are software metrics being used regularly? 1 - No 2 - Unsure how metric information is being applied 3 - Metrics play a key role in shaping maintenance decisions 13. How will metrics be used on the reengineering project? 1 - Will not be used 2 - Metrics still being defined or unsure of metrics usage 3 - Metrics will determine whether the reengineering project was successful

21. How will the reengineered software be validated?

1 - Not considered this
2 - Will use current validation test suite
3 - Detailed test plan, including functional traceability

22. Does the new maintenance environment include testing?

1 - No
2 - Testing is included but is mostly ad hoc or blackbox
3 - Testing is an integral part of the new maintenance environment

Tools Analysis

23. Are tools available (to automate the various reengineering strategies) for the target hardware platform and source code language?

1 - No, or don't know
2 - Yes, but would require extensive customization
3 - Yes

Training

24. How much current training/understanding of reengineering is there within the maintenance organization?

1 - Few, if any, of the reengineering team have been trained
2 - Some of the reengineering team have been trained
3 - Most of the key reengineering team have been trained in reengineering, tools, and the target maintenance environment

25. Who is to be trained and in what area?

1 - Training has not been considered
2 - Training is part of this reengineering project but not in detail
3 - People have been designated and specific classes identified

26. Is there a funded plan to train the reengineering staff in any new tools or techniques which will be used?

1 - No
2 - Yes, but less than 10% of reengineering budget
3 - Training represents 10% or more of the entire reengineering budget

SOFTWARE IMPORTANCE

1. How much does the candidate software contribute to the customer's mission?

   1 - Minimally
   2 - Somewhat
   3 - Significantly

2. If the candidate software failed what would be the effect?

   1 - Little or no damage
   2 - Significant damage
   3 - Permanent damage, major financial loss, or loss of life

3. Is there a contingency plan (current, recently tested, and ready at a moment's notice) which could be used if the candidate software fails?

   1 - Yes, or not needed
   2 - Yes, but with some difficulty and a significant loss of efficiency
   3 - No

4. If the candidate software failed, would it effect other mission-critical systems or data files?

   1 - Little or no effect on other mission-critical systems or data files
   2 - Significant disruption to other mission-critical systems or data files
   3 - Permanent damage, major financial loss, or loss of life due to disruption of other mission-critical systems or data files

5. Is the candidate software important to the organization's management?

   1 - No
   2 - Somewhat, or don't know
   3 - Yes

6. Is the candidate software important to the customers?

   1 - No
   2 - Somewhat, or don't know
   3 - Yes

7. Is the software being used?

   1 - No, the staff has developed small systems or manual processes to replace the software's function
   2 - To some extent, but there are valid complaints about the software's functional shortcomings
   3 - Yes, the software performs the business functions it was meant to fulfill

	8. Does the candidate software have a maintenance backlog? 1 - No 2 - Yes, but steady or decreasing 3 - Yes, and increasing 9. Characterize the annual change traffic* of the candidate software within the last 12 months? 1 - Small or insignificant 2 - Moderate 3 - Large or significant * Annual Change Traffic is defined as "the fraction of the software product's source instructions which undergo change during a year, either through addition, deletion, or modification."

REDOCUMENT

1. The candidate software's documentation is best characterized as:

   1 - Complete and current
   2 - Incomplete or not current
   3 - Non-existent or misleading

2. If the candidate software changes, is the documentation updated after each change?

   1 - Always
   2 - Sometimes
   3 - Rarely or never

3. Is the technical documentation useful and used during maintenance activities?

   1 - Yes, the maintenance staff can easily use the documentation to understand the candidate software
   2 - Using the documentation can be time-consuming
   3 - No, the documentation is non-existent or cumbersome and difficult to use

4. Does the documentation follow a standard format (such as MIL-STD-2167A or organization-specific)?

   1 - Yes
   2 - Yes, but standard is vague or documentation does not completely comply
   3 - No standard

5. Throughout the lifetime of the candidate software, has there always been an effective enforcement process to ensure adherence to the organization's documentation standards?

   1 - Yes
   2 - The enforcement process is not consistently applied or monitored
   3 - No enforcement process

TRANSLATE SOURCE CODE

1. What is the candidate software's principal language level?

   1 - Advanced, high level languages such as Ada, C++, Focus, etc.
   2 - Higher order languages such as FORTRAN, COBOL, C, Pascal, etc.
   3 - Assembly languages

2. How many programming languages does the candidate software use?

Note: Embedding SQL or similar database language within a High Order Language does not count as two separate languages. Rationale: Using embedded database calls does not increase the need to reengineer the program.

   1 - One
   2 - Two
   3 - Three or more

3. Is there a need to translate/consolidate the source code language(s)?

   1 - No need
   2 - Not yet but anticipate a mandate soon
   3 - Strong need due to external mandate (such as Ada) or to satisfy reengineering goals

4. Is there personnel trained in the new target language?

   1 - No
   2 - Not currently but there is a defined, formal plan to hire/train personnel
   3 - Yes

5. Is the language currently used by the candidate software based on a proprietary compiler?

   1 - The compiler isbased on a DoD-approved high order language (HOL) with strict acceptance standards
   2 - The compiler has had some minor deviations/enhancements from an approved HOL
   3 - The compiler has had major modifications deviating from an approved HOL, making the compiler highly proprietary

6. Are sufficient support tools available for the language currently used by the candidate software?

   1 - Yes, for the current language there exist configuration management tools, development tools, maintenance tools, analysis tools, and a variety of compilers
   2 - There exist a limited number of support tools for the current language. Some of these may have been developed in-house
   3 - No, there are very few support tools for the existing language

	7. How portable is the language currently used by the candidate software? 1 - Language is very portable 2 - Portable, but significant embedded routines that are not portable 3 - Not very portable to other platforms 8. Is the language currently used by the candidate software mostly written in an obsolete dialect of that language? 1 - The language is the most current dialect 2 - Mixture of dialects of same language is used by the candidate software 3 - The language uses an obsolete dialect

DATA REENGINEER

1. Characterize the data files:

   1 - Data is managed by a modern data file organization (such as relational or object-oriented databases)
   2 - Some of the data is managed by a modern data file system, but other files use old formats (such as flat files, VSAM, etc.)
   3 - Data is mostly managed by old file formats (such as flat files, VSAM, etc.)

2. Are the relationships between data elements well-documented?

   1 - Yes
   2 - Not consistently
   3 - Data relationships not clearly defined or understood

3. How old are the current data files?

   1 - Less than 3 years
   2 - Three to five years
   3 - More than 5 years

4. How many different data files?

   1 - One to three
   2 - Three to five
   3 - More than five

5. Is the candidate database hindering migration to more advanced technology?

   1 - No
   2 - Some data files create a problem
   3 - Yes

6. Are the data files used by other systems?

   1 - Yes
   2 - Yes, but only one or two files are shared with other systems
   3 - No, the files to be data reengineered are exclusively used by a single system

7. Does the software maintenance organization change modules or just develop new ones when functionality changes? (bearing on data name rationalization)

   1 - Maintenance usually changes existing modules
   2 - Maintenance sometimes changes existing modules and sometimes creates new modules
   3 - Maintenance usually creates new modules

	8. Does the organization use a standard data dictionary? 1 - Yes 2 - Yes, but not strictly enforced and therefore not current 3 - No standard data dictionary 9. How many different programs within the candidate software? 1 - One to three 2 - Three to seven 3 - Over seven 10. If any recently reengineered/redeveloped software uses the candidate data file, what is the impact to the candidate data file? 1 - Insignificant or no impact to candidate data file 2 - Minimal impact to candidate data file 3 - The candidate data file is critical to the reengineered software and will be significantly affected by the software reengineering project

RETARGET

1. How portable is the language currently used by the candidate software?

   1 - Language is very portable
   2 - Portable but significant embedded routines that are not portable
   3 - Not very portable to other platforms

2. Is the candidate software stand-alone or is it a part of a larger system?

   1 - Candidate software is closely connected to a larger system
   2 - Candidate software is connected to a larger system but candidate software could be isolated with a well-defined interface
   3 - No, candidate is a stand-alone

3. What is the age of the current hardware platform?

   1 - Less than 3 years old
   2 - Three to six years old
   3 - Over six years old

4. Does the organization need to maintain duplicates of the same candidate software due to different hardware platforms required for the software to execute upon?

   1 - No
   2 - Not sure, or differences are not significant and easily maintainable
   3 - Yes

5. Does the new maintenance environment call for CASE tools?

   1 - No
   2 - Not currently, but will probably move to CASE tools soon
   3 - Yes

6. Does the current hardware platform cause severe execution problems?

   1 - No
   2 - Yes, but not frequently
   3 - Yes

7. Is vendor support for the current hardware guaranteed for the remaining software life?

   1 - Yes
   2 - No, but support loss is unlikely
   3 - No, and this is an immediate concern

	8. Is the current hardware platform limiting expansion or future development activities? 1 - No 2 - To some extent 3 - Yes, further software expansion, future development, or new technical advances are limited due to the existing hardware platform

	RESTRUCTURE
	1. What is the average number of Source Lines of Code (SLOC) per Computer Software Unit (CSU*) in the candidate software? * A CSU is defined as the smallest cohesive software unit that is separately testable. A CSU usually corresponds to a called function or sub-routine within a program. 1 - Less than 50 2 - 50 and 200 3 - More than 200 2. What is the average number of Function Points per CSU of the candidate software? 1 - Less than 500 2 - 500 - 2500 3 - More than 2500 3. What is the average Cyclomatic complexity per module? 1 - Ten or less 2 - Between 10 and 20 3 - More than 20 4. What is the average Essential complexity per module? 1 - Five or less 2 - Between 5 and 10 3 - More than 10 5. The candidate software was created using a development process that was: 1 - Rigidly followed 2 - Sometimes followed 3 - Not followed or non-existent 6. The change control procedure for the candidate software has been: 1 - Strictly enforced 2 - Loosely enforced 3 - Ad hoc or does not exist

	7. Characterize the annual change traffic* of the candidate software within the last 12 months? 1 - Small or insignificant 2 - Moderate 3 - Large or significant * Annual Change Traffic is defined as "the fraction of the software product's source instructions which undergo change during a year, either through addition, deletion, or modification." 8. Does software maintenance usually involve changing modules or developing new modules? 1 - Maintenance usually is implemented by creating new modules 2 - Maintenance uses approximate equal new modules vs. changing existing modules 3 - Maintenance usually consists of changing existing modules 9. Is technology available for the analysis of the candidate software (complexity, function point analysis, etc.)? 1 - No 2 - Don't know, or not entirely accessible 3 - Yes 10. Over the past two years, does the candidate software require more man-hours to modify per SLOC? 1 - Maintenance man-hours per SLOC is low and constant 2 - Maintenance man-hours per SLOC is increasing slowly 3 - Maintenance man-hours per SLOC is increasing significantly

	MAINTENANCE ENVIRONMENT
	1. Will the size of the candidate software's support staff remain stable? 1 - Yes 2 - Minor staff changes are anticipated 3 - Major staff changes/cutbacks are due 2. For the candidate software, are there sufficient maintenance programmers? 1 - Yes, allowing timely maintenance and happy users 2 - Current number of maintenance personnel is usually sufficient with occasional maintenance backlogs 3 - No, thus causing a severe maintenance backlog and unhappy users 3. Is there sufficient maintenance personnel with in-depth experience on the candidate software? 1 - Yes 2 - Yes, but may be losing some in the near future 3 - No 4. Characterize the maintenance personnel turnover (per year): 1 - Very stable group of maintenance personnel 2 - A few people leave each year 3 - Heavy or frequent turnover 5. Are the original developers available for consultation? 1 - Yes 2 - Yes, but the candidate software is old or the developers are not easily accessible 3 - No 6. If maintenance personnel for the candidate software leave, will they be replaced? 1 - Yes 2 - Some natural attrition is a goal of the organization 3 - No 7. Is the candidate software more expensive (man-hours, productivity loss, etc.) to maintain when compared to the rest of the software maintained by the organization? 1 - Less expensive than average 2 - About average 3 - More expensive than average

	8. How do the users view the candidate software's quality? 1 - Improving orsatisfactory 2 - Remaining the same and barely tolerable 3 - Declining 9. Characterize the Mean Time Between Fixes (MTBF) for the candidate software (i.e. how often is a software error discovered): 1 - Few if any errors are detected 2 - Moderate number of errors 3 - Frequent errors with more created after each modification 10. After an error is detected, characterize the Mean Time To Repair (MTTR) the candidate software: 1 - Errors are fixed quickly 2 - Errors are added to the backlog and fixed within a reasonable amount of time 3 - The backlog requires a large amount of time before an error is addressed

Reengineering Technical Assessment
Table 4-2 RTA Summary Worksheet

Candidate Software Name & Description: ________ ________ ________	**Number of Questions Answered**	**Summation of Answers**	**Average: (Summation Divided by Number of Questions)**	**Indication of Strategy**
Remaining Life (>3 yrs?)				yes no
Age (> 5 yrs?)				yes no
Preparation[1]				yes no
Importance[1]				yes no
Maintenance Environment[1]				yes no
Redocument[2]				yes maybe no
Restructure[2]				yes maybe no
Translate Source Code[2]				yes maybe no
Data Reengineer[2]				yes maybe no
Retarget[2]				yes maybe no
Reverse Engineer				yes no
Redevelop				yes no
Status Quo				yes no

[1] The “yes” range is from 2.00 to 3.00. The “no” range is below 2.00.

[2] The “yes” range is from 2.40 to 3.00. The “maybe” range is from 1.70 to 2.39. The “no” range is below 1.70.

5. Reengineering Economic Assessment Process

5.1 Introduction

The Reengineering Economic Assessment (REA) process is the second part of the Software Reengineering Assessment (SRA) process. The REA process is performed on candidate software identified in the Reengineering Technical Assessment (RTA) process (Section 4). An REA is done on all the technically appropriate strategies being considered for a specific candidate. If more than one candidate is being evaluated, the REA is repeated for each candidate. Finally, in the Reengineering Management Decision (RMD) Process (Section 6) the results of all the RTAs and REAs are re-evaluated in terms of programmatic, organizational, and other non-economic issues to establish overall organizational priorities.

The REA is intended to be performed by one or more software analysts with an understanding of software economics. These personnel need to have a thorough understanding of the existing software, reengineering, cost estimation/modeling, and economic indicators, or have access to someone who has this knowledge. The purpose of the REA is to develop credible and consistent cost estimates for each strategy.

Although highly accurate cost estimates are always desirable (and the REA makes every attempt to display all costs accurately), the REA's focus is on the consistency of the estimates for purposes of decision making. The accuracy required is that which produces the same results as an assessment performed by one or more independent analysts. The preferred economic indicators used, Net Present Value (NPV) and Benefit Investment Ratio (BIR), are intended for comparing strategies on equal terms rather than producing a budgetary input. After selected strategies are identified and organizational priorities are established, detailed budget quality estimates can be developed in current year/then year (inflated) dollars for each proposed effort.

Finally, the REA is not intended to be a comprehensive tutorial on software estimating or economics, nor does it produce cost estimates that can be used to support a budget or Defense Acquisition Board/Major Automated Information System Review Committee/Council (DAB/MAISRC) Review.

5.2 Overview

A general overview of the REA process is illustrated in Figure 5-1. Ground rules and assumptions are identified and provide management flexibility. A common Cost Element Structure (CES) is developed that captures all the costs associated with all the strategies. Estimating methods are identified, necessary data is collected, and estimates for each cost element are developed. The estimates are aggregated, converted into their Present Value, and used to calculate the NPV and BIR for each candidate strategy. Other economic indicators, such as Break-even Point (BP) and Rate of Return (ROR) can be used, but generally require additional effort. Sensitivity and risk analyses may be performed to identify the sensitivity and risk associated with the estimates.

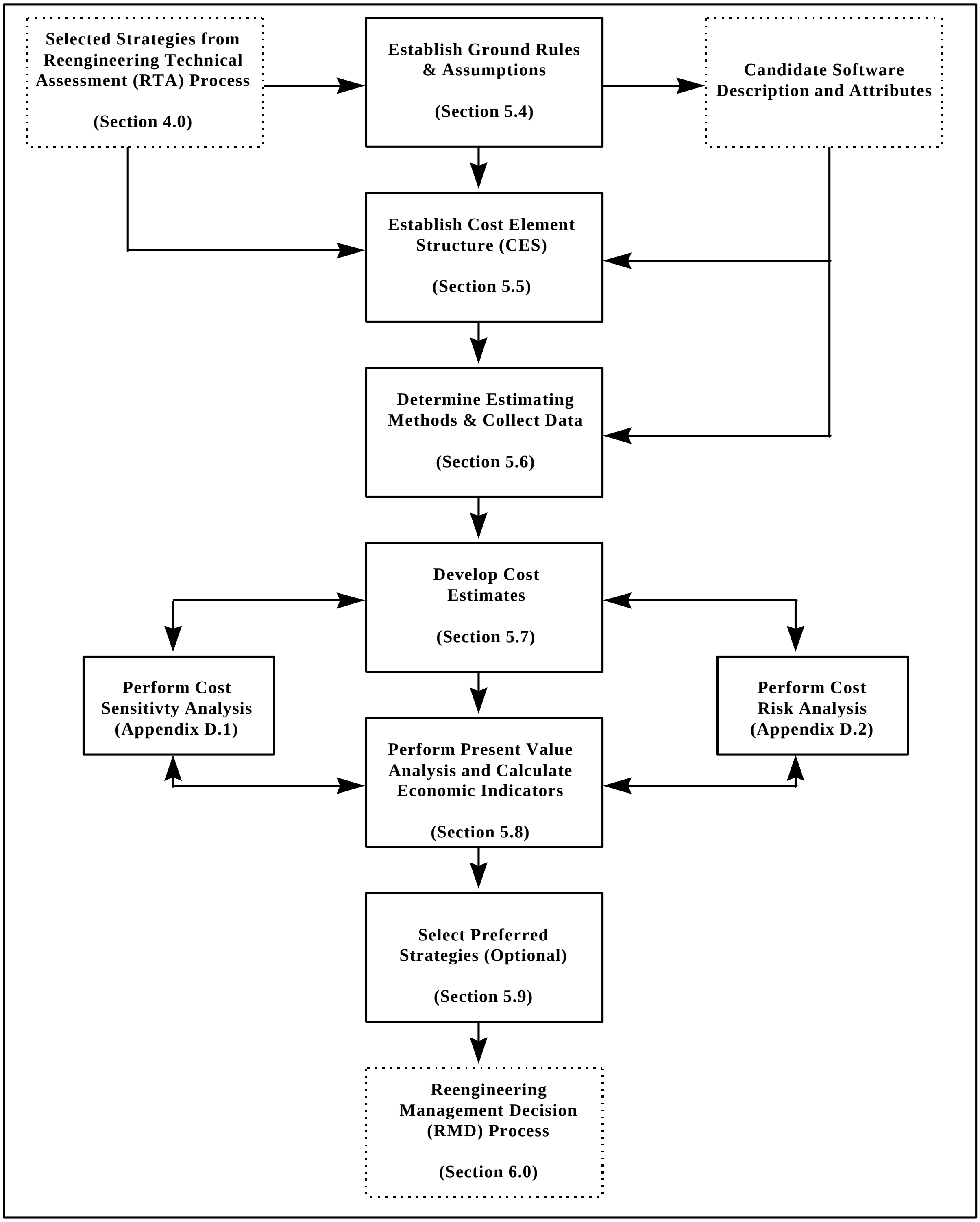


Figure 5-1. Reengineering Economic Assessment (REA) Process Overview

The estimates of specific cost elements for each strategy should be based on the best information available. Vendor quotes or catalog prices are typically better sources than parametric estimating models or engineering judgment. However, when a parametric software estimating model is used to estimate the costs of specific cost elements, it is imperative to use the same model for the estimates of the same cost element for all the other strategies.

The estimate for each strategy should be made at the appropriate level. This can be at the system level, the Computer Software Configuration Item (CSCI) level for embedded systems, or the computer program level for MIS systems. The CSCI or program level is generally the most appropriate level. Estimates at the system level should be avoided if possible because they may be at too gross a level and thus inaccurate. Estimates are not generally made below the CSCI level (at the computer software unit (CSU) level or computer software component (CSC) level), unless this is where the specific need is, because they may be at too detailed a level and require unnecessary effort.

5.3 Step-by-Step Approach

This section provides a simplified description of each of the major steps required to accomplish an REA. The reader is referred to sections 5.4 through 5.9 for supporting details and theory. The objective of the economic assessment is to prepare a cost estimate for each cost element by fiscal year using constant dollars. The output of the REA is a report that includes the economic information required for the Detailed Assessment Results (DAR) worksheet as shown in Table C-1.

Step 1: Establish Ground Rules & Assumptions (Refer to Section 5.4): Ground Rules and Assumptions (GR&As) are established and documented to provide a consistent basis for all estimates. Figure B-1 (found in Appendix B) is an example of a GR&As Worksheet that can be tailored for use on a specific project.

Step 2: Establish Cost Element Structure (Refer to Section 5.5): A common Cost Element Structure (CES) is established for estimating all candidates and strategies. A CES is similar to a work breakdown structure. Figure 5-2 is an example of a generic CES. The CES should include cost elements that capture all costs associated with acquiring, developing, and/or maintaining all strategy-specific software, hardware, software engineering tools, training, and other elements. Only costs incurred from the beginning of the reengineering effort to the end of the software's useful life are included. Sunk costs are not included. Investment costs are usually not required for the Status Quo (Strategy 1). Costs that are identical for all strategies (referred to as wash costs) must be included for all strategies. Costs for Operations & Support (O&S) of the Status Quo during the reengineering period (referred to as phase-out costs) must be included in the O&S costs for all reengineering strategies.

Step 3: Determine Estimating Methods & Collect Data (Refer to Section 5.6): Estimating methods are determined for each cost element. Methods that are typically used include level of effort, catalog prices or written quotations, engineering estimates (bottom-up), expert opinion, analogy, and/or parametric models. Once an acceptable estimating method is identified for each CES cost element, the analyst must collect the data necessary to actually perform the estimating calculations. The analyst should strive to obtain the most accurate, precise, current, and applicable prices available. However, good judgment should be used in determining the amount of effort expended to obtain the best prices. In order to develop an REA in a timely manner, engineering estimates can be used in place of prices that are difficult and time consuming to obtain.

Step 4: Develop Cost Estimates (Refer to Section 5.7): The worksheets shown in Appendix B can be used to consolidate the various cost element estimates. Once the annual costs in constant-dollars for all CES elements have been determined for each strategy, the results can be aggregated to the appropriate higher level (CES 1.0, 2.0 or 3.0) annual totals. Simple multiplication by the appropriate discount factors will convert each annual total to its present value. When using parametric models, the analyst should use the same model for all strategies and candidate software that will be compared. Parametric models may need to be adjusted for reengineering versus new development.

Step 4.1: (Optional): Perform Cost Sensitivity Analysis (Refer to Appendix D.1): The analyst may perform cost sensitivity analyses on the primary variables used by parametric estimating models to measure how sensitive the estimate is to changes in the variable. This typically involves iteratively applying the model while varying the value of the parameter being evaluated.

Step 4.2: (Optional): Perform Cost Risk Analysis (Refer to Appendix D.2): The analyst may perform cost risk analysis to adjust the estimates for each strategy to the same confidence level.

Step 5: Perform Present Value Analysis and Calculate Economic Indicators (Refer to Section 5.8): The analyst calculates the NPV and BIR of each strategy. This is done by using the definitions provided in Section 5.8. Other economic indicators may be used to determine the most cost effective strategy including Break-even Point (BP) and Rate of Return (ROR). The present value of total cost and present value of total benefit may also be considered. The bottom line numbers from the cost worksheets are entered into the appropriate cells in the DAR worksheet (Table C-1). The REA analyst meets with the RTA team to co-author the Management Report as described in Section 6.

Step 6: (Optional) Select Preferred Strategies (Refer to Section 5.9): The analyst may economically rank-order the proposed strategies using an appropriate economic indicator prior to meeting with the technical personnel for the reengineering management decision process.

5.4 Establish Ground Rules and Assumptions (GR&As)

All estimates are based on the same Ground Rules and Assumptions (GR&As). A worksheet containing examples of typical ground rules and assumptions is provided in Figure B-1. These should be tailored to each specific project. GR&As provide part of the framework upon which the estimate is developed. They describe how technical issues relate to the estimating process and state how specific issues will be treated. They also provide a method for addressing uncertainty by permitting the analyst to assume a response to a required input or factor when the actual value or response is unknown. These must be specifically stated so that the decision maker understands the basis for the estimate. For example, some parametric software estimating models may include a variable for which the analyst does not have an input. Frequently this can be resolved by making an assumption based on professional judgment and a thorough understanding of the program being estimated. Also, your organization may have a set of organizational default values or criteria for choosing these values.

5.5 Establish Cost Element Structure (CES)

The REA for each candidate and strategy being considered must be based on a common Cost Element Structure (CES) for all candidates and strategies. A generic CES for software is provided in Figure 5-2 and in Appendix B based on *MIL-HDBK-171, Work Breakdown Structure for Software Elements*. A detailed CES applicable to a Major Automated Information System (MAIS) is provided in the *USA Economic Analysis Manual*, Appendix D. Investment (CES 1.0) is defined as the effort required to reengineer the software. Investment costs are usually not required for Status Quo. If any, they should be included in O&S for Status Quo (CES 3.0). O&S includes operations, support (maintenance), training, and other elements for the remaining life of the software. The CES for O&S is essentially the same as for Investment since the process is similar, but on a smaller scale. The O&S cost for reengineered software includes support of the reengineered software (CES 2.0) as well as continued support (phase-out) of the Status Quo (CES 3.0) during the reengineering period.

Using a common CES will ensure that all costs for each strategy are considered in a complete and consistent manner. The CES should include cost elements that capture all costs associated with acquiring, developing, and/or maintaining all strategy-specific software, hardware, software engineering tools, training, and other elements. Hardware costs can drive the need to reengineer even though the software maintenance costs are low. Systems with high or potentially high hardware maintenance cost may need to be reengineered because the hardware is too expensive to maintain.

1.0 Investment for Reengineered Software
- 1.1 Software Development
  - 1.1.1 Requirements Analysis
  - 1.1.2 Preliminary Design
  - 1.1.3 Detailed Design
  - 1.1.4 Code & Unit Test
  - 1.1.5 Unit Integration & Test
  - 1.1.6 CSCI Test
  - 1.1.7 SPCR Resolution
  - 1.1.8 Other
- 1.2 CSCI-CSCI Integration & Test
- 1.3 System Integration & Test
- 1.4 Training
- 1.5 Data
- 1.6 Peculiar Support Equipment
- 1.7 Operational Site Activation
- 1.8 Facilities & Utilities
- 1.9 Hardware
- 1.10 System Operations
- 1.11 Independent Verification & Validation (IV&V)
- 1.12 System Engineering/Program Management
- 1.13 Other

2.0 Operations & Support for Reengineered Software
- (Repeat same sub-elements as 1.0)
- 2.14 O&S for Status Quo during Reengineering

3.0 Operations & Support for Status Quo
- (Repeat same sub-elements as 1.0)

Figure 5-2. Generic Cost Element Structure (CES) for Software Reengineering

Costs that are identical for all strategies (referred to as wash costs), must still be included for each strategy. The reason for including "wash costs" is that the cost estimates developed for the decision maker are usually translated into current year, budgetary documents and should reflect all elements which require funding. Sunk costs (past expenditures or irrevocably committed costs which are unavoidable) are not included. Costs for O&S of the Status Quo during the reengineering project (typically known as phase-out costs) must be included in the O&S costs for all reengineering strategies.

In addition to tailoring the CES for the specific costs to be estimated, the CES may be tailored to satisfy any of the following requirements:

- To match the CES from a useful and earlier estimate for this program,
- To correlate the CES output from the cost estimating model used, or
- To highlight a particular cost sensitivity, e.g., to provide more detailed information.

Further guidance on development of a CES can be found in the references identified in Section 2.0, paragraphs 2.1.1 a–c and 2.1.2 b, c, and f.

5.6 Determine Estimating Methods and Collect Data

Complete software development and support estimates typically use a variety of estimating methods including level of effort, catalog prices or quotations, engineering (bottom-up), expert opinion, analogy, and/or parametric models. Different estimating methods are typically used for different CES elements. For example, when the specific items are relatively well known or defined, vendor contract prices or quotations can frequently be used to estimate the cost of purchased training (CES 1.4), software development tools (CES 1.6), and hardware (CES 1.9). Analogies can be used when the specific items are unknown, but the relative costs for similar items are known from a similar prior effort.

Software development effort (CES 1.1) and related efforts (CES 1.2, 1.3, etc.) are typically estimated using parametric models, but may also be estimated based on analogy (in-house historical data) or best engineering estimates. The analyst doing the REA has several parametric models from which to choose. These include but are not limited to: CHECKPOINT, PRICE-S, SEER-SEM, SLIM, and SOFTCOST-OO. SASET was determined to be not applicable to reengineering.

All of these models are flexible enough to estimate the range of strategies (Status Quo and various reengineering strategies) as described in Appendices E through J. However, these models do not include all the sub-elements of CES 1.0 in their estimates. Depending on the model used, it may be necessary to estimate some of the sub-elements of CES 1.0 separately. There is no single "best" software development estimating model. The analyst is encouraged to select a model based on his/her familiarity and skill with the model, the availability of the model, the availability of input data, and the suitability of the model CES. Desired characteristics of a model include: adaptability to reengineering, ease of use, adaptability to an organization's actual historical data, accurate estimates of cost and schedule, and applicability to maintenance estimation.

Software O&S effort (CES 2.0 and 3.0) can be estimated either by using a parametric model or by performing an engineering estimate of the personnel and resources required. For existing software, the future software support effort can also be estimated by extending the actual historic support costs into the future using either a straight-line projection or by adjusting the individual annual cost estimates based on perceived changes in the support requirements.

The remaining life (YL) for all candidate strategies is the same as that established by the software operational requirements. This includes the Status Quo. Remaining life is defined as the number of years required for the Investment effort (YI) plus the number of years for the O&S effort (YS). YI is typically estimated by the model and YS is the difference between YL and YI.

$$YL = YI + YS \qquad or \qquad YS = YL - YI$$

Once an acceptable estimating method is identified for each CES cost element, the analyst must collect the data necessary to actually perform the estimating calculations. For the parametric models this includes gathering information for each of the input variables for the model. The specific values that will be used as inputs to a parametric estimating model are typically identified both from the information required to perform the RTA and from other sources familiar with or responsible for the candidate software. For other CES elements, this represents collecting vendor quotations, seeking pricing information from catalogs, and working with engineers to develop "bottom-up" engineering estimates. The analyst must have either a thorough understanding of the system or access to someone who has this knowledge, and be able to validate the data collected as to its accuracy, volatility, or appropriateness.

5.7 Develop Cost Estimates

The worksheets shown in Appendix B can be used to consolidate the various cost element estimates and economic indicators. Once the annual costs in constant dollars for all CES elements have been determined for each strategy, the results can be aggregated to the appropriate higher level (CES 1.0, 2.0, or 3.0) annual totals. Simple multiplication by the appropriate discount factor will convert each annual total to its present value. If desired, the annual costs in constant dollars can also be multiplied by the inflation rate to convert each annual total to its current (then year) value used for break-even analysis and budgeting. Additional worksheets may be prepared to vary the discount rate or inflation rate for variance analyses. The group of worksheets shown in Appendix B should be completed for each reengineering strategy being considered for a program.

Analysts might want to perform cost sensitivity analyses on the primary variables used by parametric estimating models to measure how sensitive the estimate is to changes in the variable. This typically involves repeatedly applying the model while varying the value of the parameter being evaluated. A sensitivity analysis assists the analyst and decision maker in assessing changes based on programmatic factors. For example, if the remaining life potentially could vary between 8 and 10 years, then the analyst could repeat the estimate using both the 8 and 10 year remaining life assumptions to determine if this change would affect the results of the analysis while holding the other parameters constant. Appendix D, Table D-1, provides an example of performing cost sensitivity analysis. Further information on sensitivity analysis can be found in the *USA Economic Analysis Manual*.

In addition, analysts might want to perform a cost risk analysis to adjust the estimates for each strategy to the same confidence level. Risks can be managed by estimating a dollar value for each risk whether the risk is technical, schedule, or cost. *DoD's 5000.2-R* offers seven factors for analyzing risks: threat, technology, design, support, manufacturing, cost, and schedule. The purpose of risk analysis is twofold: (1) to determine the likelihood that the risk event will take place, and (2) to determine the impact of the risk event. That is, the analysts need to measure the probability that the event will occur and also measure what the impact of the event will be. A cost risk analysis uses standard risk analysis techniques but uses costs as the measure of the "impact". Although this handbook is not intended to provide a study in risk assessment, Appendix D, Table D-2, provides an example of performing cost risk analysis. Further information on cost risk analysis can be found in the *AFSC Cost Estimating Handbook* and in the *USA Economic Analysis*

Manual. Information on general risk management techniques can be found in *Software Engineering, Risk Analysis and Management* by Robert Charette and AFMC Pamphlet 800-45, *Software Risk Management*.

5.8 Perform Present Value Analysis and Calculate Economic Indicators

Since the purpose of the REA is to compare the relative future costs and benefits of each strategy, Present Value Analysis is used. This process equitably compares the value of different annual future costs and benefits in common terms. The method for calculating present value is described in Appendix B. The discount rate is dependent upon the period of analysis and employs the real (net inflation) discount rates (as opposed to the nominal discount rates which include inflation) for discounting constant dollar flows of money. The current discount rate is provided in Appendix C of OMB Circular A-94 which is available upon request from the office of Economic Policy in OMB at (202) 395-3381 or at http://www.whitehouse.gov/wh/eop/omb. An example of present value analysis and the use of economic indicators is provided in the *USA Economic Analysis Manual*, Appendix I.

The REA uses the Total Cost (TC) and Total Benefit (TB) for comparing the costs and benefits for each strategy. Total Cost includes only costs that are incurred from the beginning of the reengineering effort to the end of the software s useful life. It specifically excludes all costs incurred prior to the beginning of the effort (i.e., sunk costs). In contrast, a Life Cycle Cost Estimate (LCCE) would include the sunk costs. Excluding sunk costs, the general estimating process for developing Total Cost estimates is very similar to developing an LCCE.

The Present Value of the Total Cost (PV(TC)), and its derivative measures are used in the REA to compare the costs of alternative strategies. Similarly, the Present Value of the Total Benefit (PV (TB)) is used to compare the benefits of alternative strategies. Since Present Value is used rather than inflation adjustments, etc., the Present Value estimates prepared in the REA process cannot be used to support development of a budget or for DAB/MAISRC milestone reviews. Budgets can be prepared by using constant dollars and applying prescribed escalation factors.

The REA of candidate strategies depends on estimating and comparing the PV(TC) and PV(TB) of each strategy using a variety of economic indicators. The preferred economic indicators are NPV and BIR. PV(TC) and PV(TB) also can be used. Other economic indicators, such as BP and ROR can be used, but generally require additional effort. The economic indicators are calculated for each strategy and recorded on the DAR worksheet per Section 6.

Present Value of Total Cost (PV(TC)): The cumulative total of Investment (CES 1.0) and Operations & Support (CES 2.0 or 3.0) costs stated in terms of Present Value. It includes all costs incurred from the beginning of the reengineering effort through the end of the software s life, excluding sunk costs. For all strategies other than Status Quo (Strategy 1), Total O&S cost includes O&S of Strategy N after reengineering, $O\&S_N$, and O&S (phase-out) of Status Quo during the reengineering period, $O\&S_{1DR}$.

$$PV(TC_N) = PV(Total\ Investment_N) + PV(Total\ Operations\ \&\ Support_N)$$

$$= PV(TI_N) + PV(TO\&S_N)$$

where N is the number of the strategy.

Present Value of Total Benefit (PV(TB)): The difference between the Present Value of the Total O&S cost of Status Quo and the Present Value of the Total O&S cost of Strategy N. For all strategies other than Status Quo, Total O&S cost includes O&S of Strategy N after reengineering, $O\&S_N$, and O&S (phase-out) of Status Quo during the reengineering period, $O\&S_{1DR}$.

$$PV(TB_N) = PV(TO\&S_1) – PV(TO\&S_N)$$

Net Present Value (NPV): Within the context of the above two defined economic indicators (total cost and total benefit), NPV is the difference between the Present Value of the Total Benefit (PV(TB)) and the Present Value of the Total Investment (PV(TI)). It represents the net difference between dollars saved or avoided and dollars invested over time expressed in equivalent amounts at a common point in time.

$$NPV_N = PV(TB_N) – PV(TI_N)$$

This equation for NPV may also be expressed as:

$$NPV_N = PV(TC_1) - PV(TC_N)$$

The preferred strategy is the strategy with the highest NPV. (*USA Economic Analysis Manual*, paragraphs 5-1, 5-2a(3), and 5-3c)

Benefit Investment Ratio (BIR): The ratio of benefit to investment. This definition of BIR is consistent with DoD definitions wherein investment measures the cost of software development and related efforts (CES 1.0) and investment does not include the O&S cost for either Status Quo or Strategy N. (*USA Economic Analysis Manual*, paragraph 5-3d)

$$BIR_N = \frac{PV(TB_N)}{PV(TI_N)}$$

Break-even Point (BP): The break-even point where the total cost of one strategy is equal to the total cost of the other. Also, where the Benefit is equal to the Investment. (*USA Economic Analysis Manual*, paragraph 5-3b)

BP = that point in time at which:

$$TC_1 = TC_N \text{ or } TB_N = TI_N$$

in current (inflated) dollars

Rate of Return (ROR): The interest rate at which the Present Value of the Total Benefit is equal to the Present Value of the Total Investment (i.e., NPV = 0 and BIR = 1.0) through the remaining life of the strategy being evaluated. (*USA Economic Analysis Manual*, paragraph 5-3f)

$$ROR = i \text{ at which } PV(TI_N) = PV(TB_N) \text{ or}$$

$$\frac{\textit{Investment (year 1)}}{(1+i)} + \frac{\textit{Investment (year 2)}}{(1+i)^2} + \ldots = \frac{\textit{Benefit (year 1)}}{(1+i)^1} + \frac{\textit{Benefit (year 2)}}{(1+i)^2} + \ldots$$

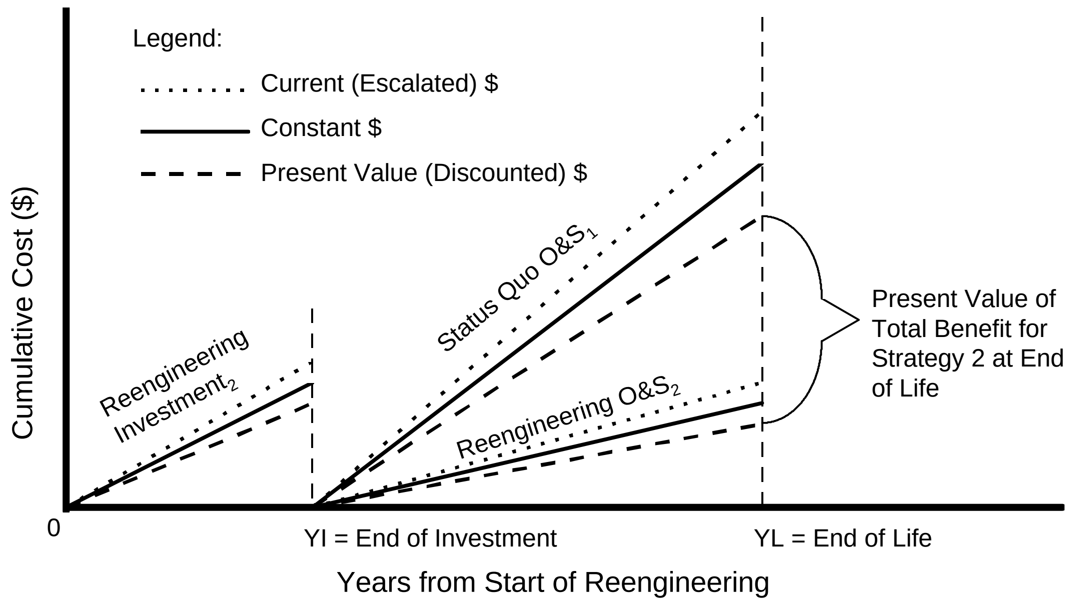
The above economic indicators, except BP, assume that all strategies have identical remaining economic lives. Where strategies have differing remaining lives, determine whether the longest or shortest life or some other time period is to be used as a basis for comparison, and make an adjustment for unequal life. If the shortest life is used to establish the time period of the analysis, recognize the residual values of the strategies with the longer lives in the cost computation. If the longest life is used, recognize the cost of extending the benefit-producing years of those strategies with a shorter life. Ensure that the decision maker is presented the complete and valid costs for each strategy for the entire length of the analysis. In cases where adjusting the remaining life is totally impractical, strategies with unequal lives may be compared based on equivalent (uniform) annual cost. (See Appendix B and the *USA Economic Analysis Manual*, paragraph 5-4a.)

Note that in the above equations for PV(TB) and NPV, the O&S cost for Status Quo during the reengineering period, O&S_{1DR} cancels out because it is present in the O&S cost of Status Quo and of Strategy N. Thus, the benefit begins accruing at the end of the reengineering investment period (YI), at the start of O&S of the reengineered software. The benefit is equal to the difference in O&S costs of Status Quo and Strategy N accruing from this point in time onward. This is graphically illustrated in Figure 5-3 for two hypothetical Strategies 1 and 2. NPV = 0 and BIR = 1 when the benefit equals the investment (in discounted $). Also, the break-even point occurs when the benefit equals the investment (in current $). For a strategy to be economically viable, these points must occur prior to the end of remaining life.

5.9 Select Preferred Strategies (Optional)

From a purely economic perspective, NPV is typically considered the most appropriate indicator of the "best" strategy. (*USA Economic Analysis Manual*, paragraph 5-1, 5-2a(3), and 5-3c.) It reflects the strategy that results in the largest "net benefit," and is useful when the actual size of the return from the alternative is the concern. However, the NPV method of comparison normally favors larger size strategies.

The BIR is also considered an appropriate indicator to rank order the candidate strategies (*USA Economic Analysis Manual*, paragraph 5-1 and 5-2). In contrast to NPV, BIR emphasizes the relationship of the *ratio* of the benefit to the investment. It reflects the strategy that results in the largest benefit relative to the investment made to achieve the benefit. Caution is recommended when using BIR because it can lead to invalid results if the BIR is higher but the benefits are either lower or the investment is higher than another alternative. The magnitude of the benefit and investment is obscured by the ratio.



NPV = PV (Benefit) - PV(Investment)

= PV(O&S$_1$) - PV (O&S$_2$) - PV(Investment$_2$)

> 0: Positive Return (Discounted $)

< 0: Negative Return (Discounted $)

= 0: Break-even (Current $)

BIR = PV(Benefit) / PV(Investment)

= (PV(O&S$_1$) - PV(O&S$_2$)) / PV (Investment$_2$)

> 1: Positive Return (Discounted $)

< 1: Negative Return (Discounted $)

= 1: Break-even (Current $)

Figure 5-3. Net Present Value (NPV), Benefit Investment Ratio (BIR), and Break-even Point (BP)

In contrast to the NPV and BIR, ROR is independent of the discount rate. A simple minimum test for ROR is that it be greater than the interest rate (i.e., cost of capital for a corporation or the interest rate on Treasury Bills for the Government). The BP is useful as a tie-breaker when NPVs, BIRs, or RORs are nearly equal between alternatives. The earlier the BP the better. BP is not sensitive to the inaccuracies that may occur after the BP. Finally, the Total Cost or Benefit could also be used to rank order the candidate strategies. However, they only consider the gross cost or benefit rather than the net benefit (NPV) of the strategy.

6. Reengineering Management Decision Process

6.1 Introduction and Overview

The Reengineering Management Decision (RMD) process (Figure 6-1) is the final step in the Software Reengineering Assessment (SRA) process. This section of the SRAH will address how the Reengineering Technical Assessment (RTA) and Reengineering Economic Assessment (REA) teams should present their findings to management and suggest criteria for management to prioritize and authorize software reengineering projects. The RMD process consists of the following 3 steps:

- **Step 1: Prepare Management Report (refer to Section 6.2):** The Management Report is necessary because the reengineering project decision maker is usually not one of the persons performing the RTA or REA. The data from these sections needs to be formally presented to management in a consistent and professional format. Even if the reengineering project decision maker has a significant part in the execution of the previous two processes, this report should still be generated as a record of the SRAH results for the given candidate reengineering project. This report should be written by the members of the RTA and REA teams as a joint effort, and then used to select the reengineering project(s).

- **Step 2: Reengineering Projects Selection (refer to Section 6.3):** Selecting the reengineering project(s) consists of the reengineering project and resource allocation decisions based on assessment results and organizational factors. This selection is primarily intended for management personnel.

- **Step 3: Implement and Document Decision (refer to Section 6.4):** This section suggests the kinds of documentation that should accompany the final decision and implementation of any SRAH-recommended reengineering projects. This documentation and implementation is primarily intended for both management personnel and the reengineering project team.

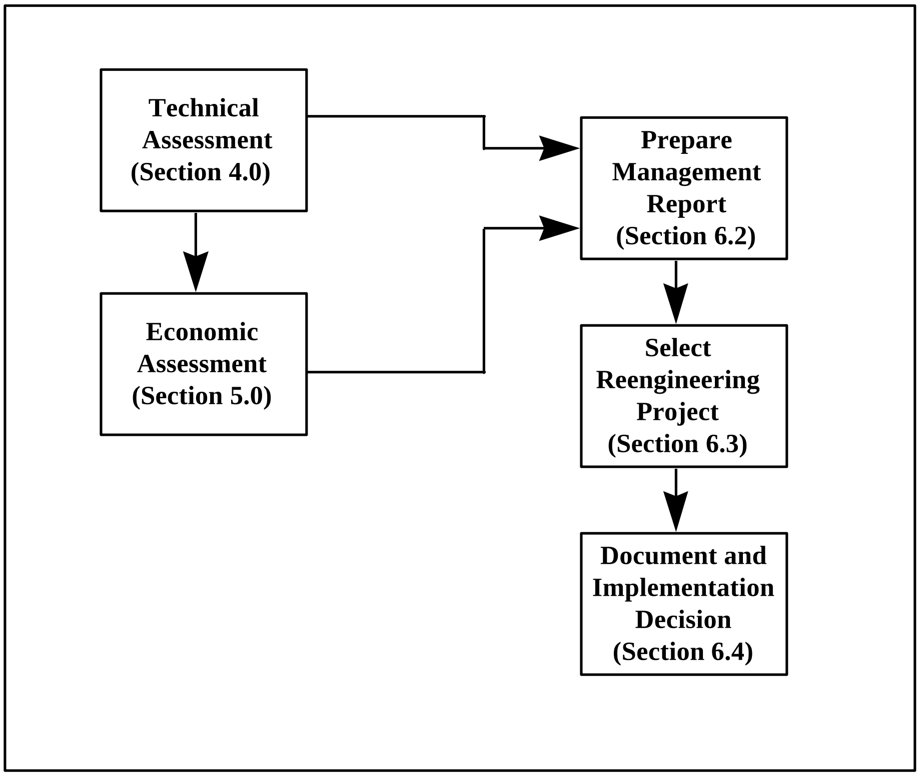


Figure 6-1. Reengineering Management Decision Process

6.2 Prepare Management Report

The Management Report consists of the following sub-sections:

- Executive Overview
- Body of the Report
- Appendices including:
  - Reengineering Technical Assessment (RTA) worksheet(s) (Results of Section 4)
  - Reengineering Economic Assessment (REA) worksheet(s) (Results of Section 5)
  - Detailed Assessment Results (DAR) worksheet(s)
  - Recommended Strategy Rank (RSR) worksheet(s)

Following completion of the RTA and REA worksheets, transfer the results to the DAR provided in Appendix C, then complete the RSR worksheet and the Executive Overview.

If another report format is desired, (i.e. if an organization has a standard managerial report format), then that alternative format can be used. The key issues are consistency and efficiency. All reports should look the same (for accurate comparison of results) and be in a format that the manager is comfortable with. The format provided in Appendix C is presented merely as a recommendation.

Detailed Assessment Results (DAR) Worksheet

Candidate Information

Strategy Information

Candidate Composite Reengineering Strategy Total (CCRS)

Candidate Software Name and Description
Special Considerations:

Strategy	Redocument	Restructure	Translate	Retarget	Data Reeng.	Reverse Eng.	Redevelop	Status Quo	Rank Within Candidate
RTA Stategy Average						Yes No	Yes No	Yes No	
Invest. Cost *								$0	
O&S Cost *								$0	
Total Cost *								$0	
Total Benefit (or ROI)								$0	
Rate of Return								N/A	
Break Even Point								N/A	
BIR								N/A	
NPV								$0	

Remain. Life	Cand. Age	Cand. Import.	Current Maint. Environ.	Reeng. Prep.

Recommended Ranking								

*Use Uniform Annual Cost when comparing strategiess or candidates of unequal life.

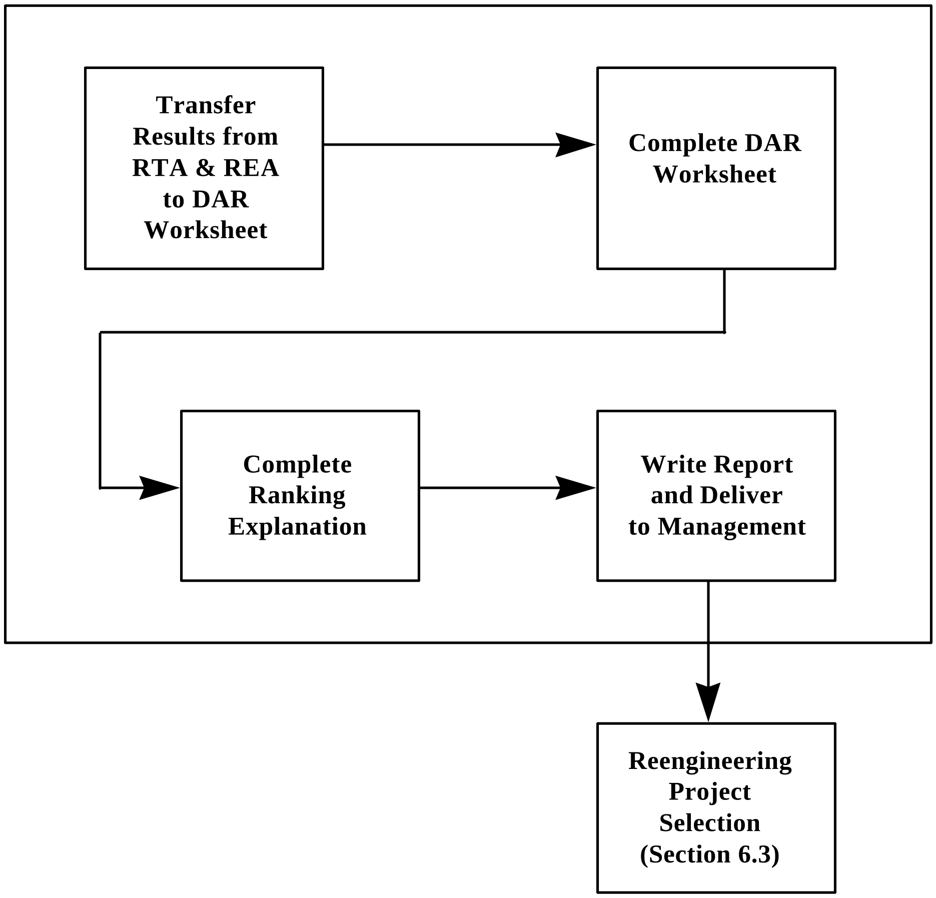


Figure 6-2. Management Report Preparation Process

6.2.1 Detailed Assessment Results (DAR) Worksheet

After completing SRAH Sections 4 and 5 (RTA and REA worksheets), the reengineering technical and economic teams should meet, with completed worksheets, to finish the 4 sections of the DAR worksheet, provided in Appendix C.

- Candidate Information
- Strategy Information
- Recommended Ranking
- Candidate Composite Reengineering Strategy (CCRS) Total

6.2.1.1 Candidate Information

Fill out the Candidate Information section of the DAR worksheet using general knowledge about the candidate software and data from the RTA worksheets. Include any special considerations that management may need to be aware of to make their reengineering decision. For example, this might include references to organizational restructuring that may affect the candidate software. Only include information that might directly affect management's reengineering decision regarding this candidate software.

Data concerning remaining life, candidate age, candidate importance, current maintenance environment, and reengineering preparation come directly from the RTA worksheets.

6.2.1.2 Strategy Information

Transfer to the DAR worksheet the economic information from the REA and RTA worksheets for each candidate software and its associated strategies. The columns and their respective units are:

- RTA Strategy Average - 3 being the highest need for this strategy, and 1 being no need (from the RTA worksheet)
- Investment Cost - dollars
- Operations and Support (O & S) Costs - dollars
- Total Cost - dollars
- Total Benefit (or ROI) - dollars
- Rate of Return - ratio
- Break-even Point - years
- Benefit Investment Ratio (BIR) - no units, but the greater the ratio, the more desirable the strategy
- Net Present Value (NPV) - dollars

6.2.1.3 Recommended Ranking

Based on the Strategy Information and Candidate Information, Rank Within Candidate prioritizes the reengineering strategies for a given software candidate. Ranking varies from 1 (high priority) to 8 (low priority). Three methods for determining this ranking are:

- **Method 1: Committee Decision**

  The most successful method for ranking strategies and candidates is to have both teams jointly rank the strategies at their meeting. Ranking should be based on the technical and economic data (particularly the RTA Strategy Average, Net Present Value and Benefit Investment Ratio), and organizational data (see Section 6.3 for list of additional considerations). This method is recommended because it has been found in SRAH field tests that the technical and economic teams have the best understanding of the candidates and thus the best perception on strategy ranking.

- **Method 2: US Army Nonquatitative Benefits Process**

  If a completely quantitative ranking method is desired (not recommended due to significantly increased DAR worksheet complexity with minimal additional benefits or increased decision accuracy), Table C-3 (SRAH page C-4 and Department of the Army, *Economic Analysis Manual*, *July 1995*) provides a method to display and compare nonquatitative benefits, (e.g., data items that are not ordinarily quantitative by nature).

- **Method 3: Existing Organizational Process**

  Use an existing methodology within the organization for ranking software maintenance requests. This method should be documented in the Executive Overview of the Management Report.

6.2.1.4 Candidate Composite Reengineering Strategy (CCRS) Total

The CCRS Total helps to compare and contrast the sum total of all reengineering strategies for a given software candidate against other software candidates. Thus, even though a single reengineering strategy may not be sufficient to warrant reengineering, a combination of reengineering strategies may favor reengineering a candidate over a candidate with a single reengineering strategy indicated.

Begin by eliminating reengineering strategies whose data should not be used in calculating the CCRS Total. A strategy should be eliminated if:

- RTA Strategy Average is less than 2.00 or “no” is indicated
- NPV is negative
- BIR is less than 1

The remaining strategy values should be totaled and entered in the corresponding CCRS Total box. This is accomplished differently for each column depending on the method used to derive that column:

- RTA Strategy Average: sum the remaining strategy values
- Investment Cost, O&S Cost, Total Cost, and Total Benefit: sum the remaining strategy values
- Rate of Return sum and Break-even Point: average the remaining strategy values
- Benefit Investment Ratio (BIR): divide the CCRS Total Benefit by the CCRS Investment Cost
- NPV: add the NPV values for the remaining strategies

6.2.2 Recommended Strategy Rank (RSR) Worksheet

When multiple software candidates and their respective reengineering strategies are being evaluated, management requires a quick way to compare the myriad possible combinations and the resulting ranking recommendations from the technical and economic teams. The RSR worksheet in Appendix C provides, on a single sheet, the key information used to rank the candidate software and their associated reengineering strategies. The information required to complete the RSR worksheet can be found on the DAR worksheets.

Software candidates and their associated reengineering strategy should be ranked high if:

- The software is important (as determined by the Importance worksheet)
- The NPV is high (as determined by the economic analysis)
- The technical strategy is highly recommended (as determined by the technical analysis)

In addition, consideration should be given to a candidate's CCRS Total. Remember, the CCRS Total (i.e. the sum total of a candidate's reengineering strategies) may indicate a candidate's ranking is higher than if only one of its reengineering strategies is considered.

Recommended Strategy Rank (RSR) Worksheet

Candidate Software Name	Recommended Strategy	Strategy NPV	Candidate Importance	Ranking

6.2.3 Finishing the Management Report

Now that the DAR and RSR (provided in Appendix C) have both been completed, it's time to write the Management Report. This will include the items listed in Section 6.2: Executive Overview, Body of the Report, and Appendices containing the RTA worksheets, REA worksheets, DAR worksheets, and RSR worksheets.

6.2.3.1 Executive Overview

The Executive Overview should consist of a single page with the recommendations of the RTA and REA teams. Take the top RSR candidates and their reengineering strategies and put them into the Executive Overview. For each top candidate and strategy, explain the reasons why it is of prime consideration. These reasons should reflect the results of the RSR: importance, technical assessment, and NPV. Any additional key reasons for selecting these top candidates and strategies should also be included and justified.

6.2.3.2 Management Report Body

The body of the report should:

- Introduce and discuss the SRAH process used to formulate the Executive Overview findings
- Discuss an overview of the current maintenance environment
- Explain why software reengineering is being considered
- Describe the candidates analyzed
- Summarize the technical considerations for each reengineering strategy chosen
- Summarize the cost estimate findings

6.3 Reengineering Project Selection

In addition to the analysis work represented by the RSR, other criteria should be considered by management. Some of this criteria is discussed in the RTA but should be emphasized to management:

- Risk Analysis and organizational strategic goals
  - What do the users/customers think of the proposed reengineering effort?
  - If implemented, how will the reengineering project affect organizational budget, schedule, other projects, and organizational strategic goals? If not implemented, how will the candidate affect mission requirements or organizational strategic competitive advantage?
  - What effect will current mission requirements have on the reengineering effort (assuming the legacy software is in use while a copy of it is being reengineered)? Can the reengineering project be successful with constantly changing mission requirements (which would dictate changing the legacy software)?
  - How important is the candidate to the organization and will that candidate remain important long enough to recover the initial reengineering investment?
  - What are the external or political issues that affect the reengineering decision (and ultimately have an effect on the reengineering project's success)?

- Resource Analysis
  - Is funding available to support the reengineering effort?
  - Are internal personnel available to perform the reengineering effort? If not, can external personnel economically and satisfactorily complete the reengineering effort?
  - Are the personnel chosen for the reengineering effort (internal or external) adequately trained and knowledgeable of organizational and reengineering goals?
  - Do the tools exist to aid the reengineering effort? If so, what does the reengineering team think of these tools, and if not, can that part of the reengineering effort be accomplished cost effectively if done manually?
  - What does the proposed reengineering team think of the proposed projects? Reengineering involves change, and some people resist change, which could result in a less than enthusiastic approach to the reengineering project (which might cause the project to fail). Successful reengineering projects have reengineering teams that embrace change and are personally ***dedicated*** to completing the reengineering project successfully. Also, an understanding of the teams' opinion on the proposed project should have a direct impact on how the project is planned and implemented.

6.4 Implement and Document Decision

After the reengineering decisions have been made and projects are authorized or denied, then an addendum to the Management Report should be written describing in detail the reasons behind the decision to reengineer or not. As mentioned above, this is needed to understand the reasoning behind the current decision for facilitating future SRAH-based reengineering decisions.

If reengineering projects are initiated following an SRAH candidate analysis, then upon completion of those projects an additional Management Report addendum should be written describing the project results. This is necessary to calibrate the SRAH process to your organization based on correlating predictions to project results.

If the decision is made to authorize a reengineering project(s), Section 2.3 contains references to reengineering preparation and project planning models. These documents can help in the implementation of the reengineering project(s) and increase the probability of reengineering success and efficiency.

Finally, if both the pre-project and post-project addendums are written and added to the Management Report, the STSC would be very interested in this project data. Such data would help to calibrate and modify future versions of the SRAH. Contact the STSC at:

Software Technology Support Center
SRAH Reengineering Survey
OO-ALC/TISEC
7278 4th Street
Hill Air Force Base, Utah 84056

(801) 775-5555 x 3054 or DSN 775-5555 x 3054
re-eng@software.hill.af.mil

APPENDIX A

REENGINEERING TECHNICAL ASSESSMENT WORKSHEET

This appendix provides a worksheet for evaluating the responses to the Reengineering Technical Assessment (RTA) questionnaire in Section 4 and identifying reengineering strategies. The answers provide the basis for identifying strategies in accordance with the selection criteria described in Section 4.

Candidate Software Name & Description: ______ ______ ______	**Number of Questions Answered**	**Summation of Answers**	**Average: (Summation Divided by Number of Questions)**	**Indication of Strategy**
Remaining Life (> 3 yrs?)				yes no
Age (> 5 yrs?)				yes no
Preparation [1]				yes no
Importance [1]				yes no
Maintenance Environment[1]				yes no
Redocument [2]				yes maybe no
Restructure [2]				yes maybe no
Translate Source Code[2]				yes maybe no
Data Reengineer				yes maybe no
Retarget [2]				yes maybe no
Reverse Engineer				yes no
Redevelop				yes no
Status Quo				yes no

Table A-1. Reengineering Technical Assessment (RTA) Worksheet

---

[1] The "yes" range is from 2.00 to 3.00. The "no" range is below 2.00.

[2] The "yes" range is from 2.40 to 3.00. The "maybe" range is from 1.70 to 2.39. The "no" range is below 1.70.

APPENDIX B

REENGINEERING ECONOMIC ASSESSMENT WORKSHEETS, PRESENT VALUE ANALYSIS, AND DISCOUNT RATES

This appendix provides five worksheets for the Reengineering Economic Assessment (REA). Figure B-1 is the Ground Rules & Assumptions (GR&As) worksheet. Tables B-1, B-2, and B-3 are worksheets for compiling and converting the various cost estimates to Present Value. The Cost Element Structure (CES) is based on *MIL-HDBK-171*, *Work Breakdown Structure for Software Elements*. Table B-4 is an aid to compute the Break-even Point. Tables B-1 through B-4 are completed for each strategy being considered. Table B-5 is the REA Summary worksheet. It is used to summarize key metrics and calculate specific economic indicators for comparing strategies. It allows for optional entry of either (1) Total Cost when the remaining life of the candidates is the same or (2) Uniform Annual Cost when the remaining life of the candidates is not the same. This appendix also provides a discussion of Present Value analysis, discount rates, and associated factors. Table B-6 contains the FY97 discount rates, mid-year discount, and factors for different periods of analysis.

The formula for Present Value (PV) for a given mid-year is defined as:

$$PV_n = F_n / (1+i)^{(n-0.5)}$$

where: PV_n = Present Value in year n

F_n = dollar amount of investment or benefit in year *n* in constant dollars

n = project year

i = interest rate or discount rate

The total PV for all years is the sum of the individual PVs for each year:

$$PV = \sum_{n=1}^{y} F_n / (1+i)^{(n-0.5)}$$

OMB Circular A-94 is the governing directive for the correct discount rate to use in economic analyses and associated studies. The Circular also contains an Appendix C which updates the discount rate annually and specifies the different rates for different types of analyses.

There are two different types of discount rates - nominal and real. Nominal interest rates are based on the economic assumptions from the President's Budget, are different for different periods of analysis, and include inflation. They are to be used for discounting nominal (current dollar) flows, as in lease-purchase analysis. Real interest rates are also based on the economic assumptions of the President's Budget, are different for different periods of analysis, but are net of inflation. The real rates reflect the real interest rates on treasury notes and bonds of specified maturities (in percent). The real rates are used for discounting real (constant dollar) flows, as in cost-effectiveness analysis.

For purpose of the REA, real discount rates will be used and are listed below for convenience.

3-Year	5-Year	7-Year	10-Year	30-Year
2.7	2.7	2.8	2.8	3.0

These real interest rates are expressed as percentages, and the next step is to develop a method to derive a factor that can be applied to constant dollars. The algorithm for converting the real interest rate to a mid-year discount factor is $1/(1+i)^{(n-0.5)}$, where i is equal to the interest rate expressed as a decimal and n is the sequence number of the year in question, beginning at program inception (year 1). Table B-6 provides the FY97 (through February 1997) real interest rates and mid-year discount factors for different periods of analysis.

A discount rate is essentially the government's projection of the cost of capital. There is either an interest cost to borrow the money for the investment, or there is an opportunity cost from not investing cash available.

It is assumed that all strategies compared for a candidate will have equal remaining lives, including Status Quo. Where it is impractical to adjust their remaining lives to be equal, strategies with unequal lives may be compared based on equivalent Uniform Annual Cost instead of Total Cost. (See USA Economic Analysis Manual, paragraph 5-4a.) For this case, Present Value is annualized (normalized) by dividing the Present Value of Investment, Benefit, or Total Cost (Cells T1, T2, or T3) by the sum of the discount factors (Cells D1, D2, or D3, respectively) for the applicable period of time. The annualized value is then compared between candidates of unequal remaining lives. Table B-5 provides for calculation of Uniform Annual Cost. Check the appropriate line at the top of the table and enter the uniform annual values in place of the total values.

Initial Ground Rules and Assumptions:

☐ All estimates and comparisons will be made in discounted constant FY ___ dollars. (Fill in the blank to specify what fiscal year the dollars estimated represent.)

☐ Use the same cost element structure (CES) for all candidate strategies.

☐ The economic "clock" (or time = 0) for any candidate strategy begins when the strategy effort starts.

☐ Sunk costs, occurring prior to the start of the effort, will not be considered.

☐ A standard labor rate of $_________/man-month (or $_________/man-year) will be used for Government, and a standard labor rate of $_________/man-month (or $_________/man-year) will be used for contractors.

☐ Strategy 1 is reserved for the "status quo" candidate strategy.

☐ Operations & Support (O&S) cost for all candidate strategies includes O&S of the existing software during reengineering.

☐ The _______________ software estimating model will be used for all estimates of reengineering and support efforts. Overall assessment (maturity, level of risk in using them, and differences between models), experience with, and track record using each model are/are not attached (circle are or are not).

☐ If code translation is indicated by the technical assessment, automated code translation will/will not be used to reduce the level of manual translation that must be performed (circle will or will not).

☐ For automated code translation, it will be assumed, that ___% of the code will be successfully translated by the automated tool based on the tool developer's specifications.

☐ If translation to Ada and restructuring are both indicated, the code will first be translated to Ada and then restructured using the Ada Programming Support Environment.

☐ The system documentation produced during the effort will be consistent with (full/tailored) (MIL-STD-xxx).

☐ To the maximum extent possible, automated CASE tools will be used to facilitate the effort. This includes the following tools: ....

☐ The staff that is currently maintaining the software will/will not be the same staff that performs the effort. If not, the _______________ will perform the work.

☐ The effort will/will not include IV&V.

Additional/Revised Ground Rules and Assumptions (include date):

☐

☐

Figure B-1. Ground Rules and Assumptions (GR&As) Worksheet

PROGRAM: ________________________ **REENGINEERING INVESTMENT COST (CES 1.0) WORKSHEET** PAGE ___ OF ___

STRATEGY:________________________ CONSTANT FY _____$K, DISCOUNT RATE _____%, INFLATION RATE _____% ANALYST____ DATE______

CES COST ELEMENT	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	TOTAL	DATA SOURCE/NOTES
1.1 Software Support												
1.1.1 Reqmnts. Analysis												
1.1.2 Prelim. Design												
1.1.3 Detailed Design												
1.1.4 Code & Unit Test												
1.1.5 Unit Integ. & Test												
1.1.6 CSCI Test												
1.1.7 SPCR Resolution												
1.1.8 Other												
1.2 CSCI-CSCI Integ. & Test												
1.3 System Integ. & Test												
1.4 Training												
1.5 Data												
1.6 Peculiar Supt. Equip.												
1.7 Operational Site Activ.												
1.8 Facilities & Utilities												
1.9 Hardware												
1.10 System Operations												
1.11 IV & V												
1.12 Sys. Eng./Pgm. Mgmt.												
1.13 Other												
1.0 Investment (Constant $)												
Mid-Yr Discount Factor												= Cell D1
1.0 Investment (Present Value)												= Cell T1
Interest Rate												
1.0 Investment (Current $)												

Table B-1. Reengineering Investment Cost (CES 1.0) Worksheet

PROGRAM: ____________________________ **REENGINEERING O&S COST (CES 2.0) WORKSHEET** PAGE ____ OF ____

STRATEGY:____________________________ CONSTANT FY ______$K, DISCOUNT RATE ______%, INFLATION RATE ______% ANALYST_____ DATE_______

CES COST ELEMENT	FY___	FY___	FY___	FY___	FY___	FY___	FY___	FY___	FY___	FY___	TOTAL	DATA SOURCE/NOTES
2.1 Software Support												
2.1.1 Reqmnts. Analysis												
2.1.2 Prelim. Design												
2.1.3 Detailed Design												
2.1.4 Code & Unit Test												
2.1.5 Unit Integ. & Test												
2.1.6 CSCI Test												
2.1.7 SPCR Resolution												
2.1.8 Other												
2.2 CSCI-CSCI Integ. & Test												
2.3 System Integ. & Test												
2.4 Training												
2.5 Data												
2.6 Peculiar Supt. Equip.												
2.7 Operational Site Activ.												
2.8 Facilities & Utilities												
2.9 Hardware												
2.10 System Operations												
2.11 IV & V												
2.12 Sys. Eng./Pgm. Mgmt.												
2.13 Other												
2.14 Status Quo O&S												Table B-3, CES 3.0 for YI only
2.0 O&S (Constant $)												
Mid-Yr Discount Factor												= Cell D2
2.0 O&S (Present Value)												= Cell T2
Interest Rate												
2.0 O&S (Current $)												

Table B-2. Reengineering O&S Cost (CES 2.0) Worksheet

PROGRAM: ____________________________ **STATUS QUO O&S COST (CES 3.0) WORKSHEET** PAGE ____ OF ____

STRATEGY:____________________________ CONSTANT FY ______$K, DISCOUNT RATE ______%, INFLATION RATE ______% ANALYST_____ DATE_______

CES COST ELEMENT	FY___	FY___	FY___	FY___	FY___	FY___	FY___	FY___	FY___	FY___	TOTAL	DATA SOURCE/NOTES
3.1 Software Support												
3.1.1 Reqmnts. Analysis												
3.1.2 Prelim. Design												
3.1.3 Detailed Design												
3.1.4 Code & Unit Test												
3.1.5 Unit Integ. & Test												
3.1.6 CSCI Test												
3.1.7 SPCR Resolution												
3.1.8 Other												
3.2 CSCI-CSCI Integ. & Test												
3.3 System Integ. & Test												
3.4 Training												
3.5 Data												
3.6 Peculiar Supt. Equip.												
3.7 Operational Site Activ.												
3.8 Facilities & Utilities												
3.9 Hardware												
3.10 System Operations												
3.11 IV & V												
3.12 Sys. Eng./Pgm. Mgmt.												
3.13 Other												
3.0 O&S (Constant $)												
Mid-Yr Discount Factor												= Cell D3
3.0 O&S (Present Value)												= Cell T3
Interest Rate												
3.0 O&S (Current $)												

Table B-3. Status Quo O&S Cost (CES 3.0) Worksheet

PROGRAM: ______________________

BREAK-EVEN ANALYSIS WORKSHEET (CURRENT $)

PAGE ___ OF ___

STRATEGY:______________________

CONSTANT FY _____$K, INFLATION RATE _____%

ANALYST____ DATE______

CES COST ELEMENT	FY___	FY___	FY___	FY___	FY___	FY___	FY___	FY___	FY___	FY___	TOTAL	DATA SOURCE/NOTES
A. ANNUAL COSTS												
3.0 Status Quo O&S (CES 3.0)												Table B-3, O&S (Current $)
2.0 Reengineering O&S (CES 2.0)												Table B-2, O&S (Current $)
1.0 Reengineering Invest. (CES 1.0)												Table B-1, Investment (Current $)
Total Reeng. Cost (1.0 + 2.0)												Sum of 1.0 and 2.0 Above.
B. CUMULATIVE COSTS												
3.0 Status Quo O&S (CES 3.0)												Cumulative of CES 3.0
2.0 Reengineering O&S (CES 2.0)												Cumulative of CES 2.0
1.0 Reengineering Invest. (CES 1.0)												Cumulative of CES 1.0
Total Reeng. Cost (1.0 + 2.0)												Sum of 1.0 and 2.0 Above.
C. BREAK-EVEN POINT												
Status Quo O&S - Total Reeng. Cost												Cumulative of 3.0 - 2.0 - 1.0
Break-even Year												BP when Status Quo - Reeng. = 0

Table B-4. Break-even Analysis Worksheet (Current $)

PROGRAM: ______________________

ANALYST: ______________________

ECONOMIC ASSESSMENT SUMMARY WORKSHEET

TOTAL COST ______ OR UNIFORM ANNUAL COST ______

METRICS	REENGINEERING STRATEGIES							
	2 _______	3 _______	4 _______	5 _______	6 _______	7 _______	8 _______	9 _______
A. PARAMETERS								
A1 SLOC / FP								
A2 ESLOC/EFP								
A3 Annual Change Traffic (ACT %)								
A4 Remaining Life Years (YL)								
A5 InvesrmentYears (YI)								
A6 Support Years (YS)								
B. DISCOUNTED FY9____$								
B1 PV Total O&S (Status Quo)								
B2 PV Total O&S (Strategy N)								
B3 PV Total Benefit (TB)								
Rank Order								
B4 PV Total Investment (TI)								
Rank Order								
C. ECONOMIC INDICATORS								
C1 Net Present Value (NPV)								
Rank Order								
C2 Benefit Investment Ratio (BIR)								
Rank Order								
C3 PV Total Cost (TC)								
Rank Order								
C4 Break-even Point (Yrs)								
Rank Order								
C5 Rate-of-Return (%)								
Rank Order								

Table B-5. Economic Assessment Summary Worksheet

Period of Analysis	3-year	5-year	7-year	10-year	20-year	30-year
Discount Rate	**0.0270**	**0.0270**	**0.0280**	**0.0280**	**0.0290**	**0.0300**
Year						
1	0.9868	0.9868	0.9863	0.9863	0.9858	0.9853
2	0.9608	0.9608	0.9594	0.9594	0.9580	0.9566
3	0.9356	0.9356	0.9333	0.9333	0.9310	0.9288
4		0.9110	0.9079	0.9079	0.9048	0.9017
5		0.8870	0.8831	0.8831	0.8793	0.8755
6			0.8591	0.8591	0.8545	0.8500
7			0.8357	0.8357	0.8304	0.8252
8				0.8129	0.8070	0.8012
9				0.7908	0.7843	0.7778
10				0.7692	0.7622	0.7552
11					0.7407	0.7332
12					0.7198	0.7118
13					0.6995	0.6911
14					0.6798	0.6710
15					0.6607	0.6514
16					0.6420	0.6324
17					0.6239	0.6140
18					0.6064	0.5961
19					0.5893	0.5788
20					0.5727	0.5619
21						0.5456
22						0.5297
23						0.5142
24						0.4993
25						0.4847
26						0.4706
27						0.4569
28						0.4436
29						0.4307
30						0.4181

Table B-6. FY97 Discount Rates and Mid-Year Discount Factors for Different Periods of Analysis

APPENDIX C

REENGINEERING MANAGEMENT DECISION PROCESS WORKSHEETS

This appendix provides the worksheets for the Reengineering Management Decision (RMD) process described in Section 6. Table C-1 provides the Detailed Assessment Results (DAR) worksheet discussed in paragraph 6.2.1. Table C-2 provides the Recommended System Rank (RSR) worksheet discussed in paragraph 6.2.2. Table C-3 is a worksheet for performing Non-Quantitative Benefits Analysis discussed in paragraph 6.2.1.3, and Table C-4 is an example for a fictitious sample case. (See *USA Economic Analysis Manual*, Figures 7-6 and I-6.)

Detailed Assessment Results (DAR) Worksheet

Candidate Information

Candidate Software Name and Description
Special Considerations:

Remain. Life	Cand. Age	Cand. Import.	Current Maint. Environ.	Reeng. Prep.

Strategy Information

Candidate Composite Reengineering Strategy Total (CCRS)

Strategy	Redocument	Restructure	Translate	Retarget	Data Reeng.	Reverse Eng.	Redevelop	Status Quo	Rank Within Candidate
RTA Stategy Average						Yes No	Yes No	Yes No	
Invest. Cost *								$0	
O&S Cost *								$0	
Total Cost *								$0	
Total Benefit (or ROI)								$0	
Rate of Return								N/A	
Break Even Point								N/A	
BIR								N/A	
NPV								$0	

Recommended Ranking

	Redocument	Restructure	Translate	Retarget	Data Reeng.	Reverse Eng.	Redevelop	Status Quo

*Use Uniform Annual Cost when comparing strategiess or candidates of unequal life.

Table C-2. Recommended Strategy Rank (RSR) Worksheet

Candidate Software Name	Recommended Strategy	Strategy NPV	Candidate Importance	Ranking

Table C-3. Comparison of Nonquantitative Benefits

Comparison of Nonquantitative Benefits							
Benefit Attribute	Weight	Alternative 1 (Status Quo)		Alternative 2		Alternative 3	
		Rank	Score	Rank	Score	Rank	Score
Total Score							

Table C-4. Comparison of Nonquantitative Benefits, Example

Comparison of Nonquantitative Benefits							
Benefit Attribute	Weight	Alternative 1 (Status Quo)		Alternative 2		Alternative 3	
		Rank	Score	Rank	Score	Rank	Score
1. Reliability	5	2	10	3	15	1	5
2. Security	5	1	5	3	15	3	10
3. Compatibility	4	3	12	1	4	2	8
4. Morale	3	3	9	1	3	2	6
5. Quality	5	3	15	1	5	2	10
6. Adaptability	3	1	3	2	6	3	9
7. Productivity	1	3	3	2	2	1	1
8. Availability	4	1	4	2	8	3	12
9. Versatility	2	1	2	2	4	3	6
Total Score			63		62		67

APPENDIX D

COST SENSITIVITY ANALYSIS AND COST RISK ANALYSIS

This appendix provides information on how to perform a cost sensitivity analysis and a cost risk analysis.

D.1 Cost Sensitivity Analysis

This section provides general guidance for assessing the sensitivity of an input variable for a parametric estimating model or any assumptions upon which the analysis is based which may vary. This process should be used whenever the specific value for the input parameter is uncertain but can be determined within known limits. It is different from a cost risk analysis (Section D.2). It focuses on the variability of one or more specific model parameters and the potential impact of that variable on the resulting estimate.

In the case where there is uncertainty in a major cost driver, the analyst should perform a sensitivity assessment to determine if more research is required to better assess that cost driver. For example, if there were reason to believe the actual lifetime *might* be 25 percent longer than specified, the effect of this difference may change the NPV or BIR calculations and result in a different rank ordering of the candidate strategies. The sensitivity analysis is done by simply running the model or recalculating the equation repeatedly while changing the value of the variable being evaluated.

An illustrative, hypothetical example is shown in Table D-1. This example adds another column to each of the two strategies being considered to accommodate a possible increase in the life of the software from 12 years to 15 years. It assumes a 20-year discount rate of 2.75%, an investment of $3.0M, and annual support costs of $600K for the existing software versus $300K for the restructured software. This change causes the preferred strategy (assuming a 7% return is acceptable) to change from Strategy 1, Status Quo, to Strategy 2, Restructuring. This results from the three additional years of reduced support costs available to recover the investment costs because the Break-even Point is at year 14.

D.2 Cost Risk Analysis

This section demonstrates a standard technique for assessing the *cost* risk associated with a point estimate. Cost risk should always be evaluated, especially when the program has fundamental ground rules and assumptions that include significant uncertainty. Assessing cost risk may easily influence the choice of a preferred reengineering strategy. Risk analysis should be performed on technical and schedule considerations as well as cost. Refer to the *AFSC Cost Estimating Handbook.*

Risk needs to be considered because not all candidate strategies will experience the same level of technical and program uncertainty. In general, maintaining the status quo (Strategy 1) incurs the least *cost* risk because there is a track record of actual experience. Strategies requiring extensive redevelopment have significantly more cost, schedule, and technical risk. The goal is to assess risk and to ensure that the estimates for all the strategies have the same or similar confidence level. This contributes to making the estimates more comparable and objective.

Cost risk assessment is accomplished by adding (or subtracting) an element of cost to Investment (CES 1.0) and to Operations & Support (CES 2.0) to adjust the Total Cost estimate to the same confidence level or expected value for each strategy. Nominally, the adjustment will establish an expected value of 50% estimating confidence. This is defined as the estimate (i.e., the mean) for which there is an equal chance that the final outcome will be lower or higher than the estimate. The cost risk assessment is a four-step process, as summarized in Figure D-1 and described below (*Boehm 81*, page 319):

Step 1 - Determine those areas of major schedule, performance, or cost risk.

Step 2 - For each major risk area, establish lowest possible value (L), most likely value (ML), and highest possible value (H) for each cost or cost driver. The L and H values typically represent the $\pm 3\sigma$ (three standard deviation) limits on the probability distribution of the actual software size, normally encompassing 99.7% of the cases.

Step 3 - Assume a beta probability distribution and calculate the expected value (EV).

Step 4 - Recalculate the cost using the EV. The new cost represents the new estimate at 50% estimating confidence.

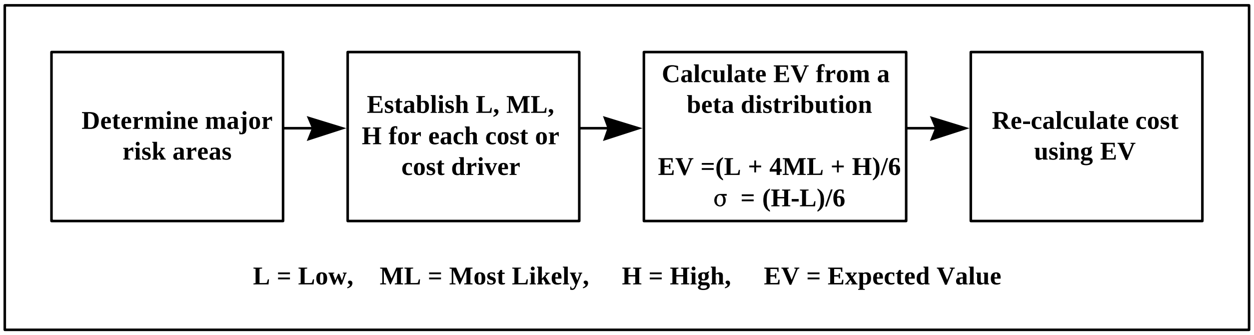


Figure D-1. Cost Risk Assessment Process

A hypothetical example is shown in Table D-2. This example accommodates an estimated 20% uncertainty (30,000 versus 25,000) in the size (SLOC) related to restructuring the software. The uncertainty over the number of SLOC prompted the analyst to recalculate the strategy estimate replacing the original ML value of SLOC with the new EV value of SLOC. For this example:

$$L = 20{,}000 \quad H = 60{,}000 \quad ML = 25{,}000$$

$$EV = (L + 4ML + H)/6 = 30{,}000$$

Using NPV or Total Cost, this new estimate causes a change in the rank order of the strategies (from 1 to 2) but assures equal confidence in all of the estimates. Notice that if BIR, Total Benefit, or Total Investment had been used to rank the strategies, the rankings would have remained unchanged. The NPV was used as the ranking metric because the overall objective was to reduce total software costs rather than obtain the biggest savings on investment.

PROGRAM: Example 1

ECONOMIC ASSESSMENT SUMMARY WORKSHEET

ANALYST: ______________________

TOTAL COST X **OR UNIFORM ANNUAL COST** ______

METRICS	REENGINEERING STRATEGIES							
	2 Restr	3 Restr	4 _______	5 _______	6 _______	7 _______	8 _______	9 _______
A. PARAMETERS								
A1 SLOC / FP								
A2 ESLOC/EFP								
A3 Annual Change Traffic (ACT %)								
A4 Remaining Life Years (YL)	12	15						
A5 InvesrmentYears (YI)								
A6 Support Years (YS)								
B. DISCOUNTED FY9____$								
B1 PV Total O&S (Status Quo)	$5,077	$6,325						
B2 PV Total O&S (Strategy N)	$2,538	$3,163						
B3 PV Total Benefit (TB)	$2,539	$3,162						
Rank Order	2	1						
B4 PV Total Investment (TI)	$2,960	$2,960						
Rank Order	N/A	N/A						
C. ECONOMIC INDICATORS								
C1 Net Present Value (NPV)	($421)	$202						
Rank Order	2	1						
C2 Benefit Investment Ratio (BIR)	0.86	1.07						
Rank Order	2	1						
C3 PV Total Cost (TC)	$5,498	$6,123						
Rank Order	1	2						
C4 Break-even Point (Yrs)	14	14						
Rank Order	N/A	N/A						
C5 Rate-of-Return (%)								
Rank Order								

Table D-1. Reengineering Economic Assessment (REA) Summary Worksheet with Cost Sensitivity Example

PROGRAM: Example 2 **ECONOMIC ASSESSMENT SUMMARY WORKSHEET** PAGE: ______ OF ______

ANALYST: ______________________ TOTAL COST _X_ OR UNIFORM ANNUAL COST ______ DATE: ________________

METRICS	REENGINEERING STRATEGIES								
	2 Redoc	3 Restr	4 #3 w/risk	5 _______	6 _______	7 _______	8 _______	9 _______	DATA SOURCE/NOTES
A. PARAMETERS									
A1 SLOC / FP									Counted or Estimated
A2 ESLOC/EFP	25000	25000	30000						Counted or Estimated
A3 Annual Change Traffic (ACT %)									Calculated or Estimated
A4 Remaining Life Years (YL)									Specified
A5 InvesrmentYears (YI)									Estimated
A6 Support Years (YS)									YS = YL - YI
B. DISCOUNTED FY9_7___$									
B1 PV Total O&S (Status Quo)	$4,537	$4,537	$4,537						Table B-3, Cell T3
B2 PV Total O&S (Strategy N)	$3,979	$2,764	$3,065						Table B-2, Cell T2 for Each Strategy
B3 PV Total Benefit (TB)	$558	$1,773	$1,472						B1 - B2 for Each Strategy
Rank Order (before)	2	1							
Rank Order (after)	2		1						
B4 PV Total Investment (TI)	$226	$947	$1,179						Table B-1, Cell T1 for Each Strategy
Rank Order (before)	1	2							
Rank Order (after)	1		2						
C. ECONOMIC INDICATORS									
C1 Net Present Value (NPV)	$332	$826	293						B3 - B4 for Each Alternative
Rank Order (before)	2	1							
Rank Order (after)	1		2						
C2 Benefit Investment Ratio (BIR)	2.47	1.87	1.25						B3 / B4 for Each Alternative
Rank Order (before)	1	2							
Rank Order (after)	1		2						
C3 PV Total Cost (TC)	$4,205	$3,711	$4,244						B2 + B4 for Each Strategy
Rank Order (before)	2	1							
Rank Order (after)	1		2						
C4 Break-even Point (Yrs)									Year when Current TC(1) = TC(N)
Rank Order (before)									
Rank Order (after)									
C5 Rate-of-Return (%)									Interest Rate when NPV = 0
Rank Order (before)									
Rank Order (after)									

Table D-2. Reengineering Economic Assessment (REA) Summary Worksheet with Cost Risk Example

DOCUMENTATION REVIEW AND EVALUATION FORM

Document: *Software Reengineering Assessment Handbook*, Version 3.0

Reviewed by: ______________________________

Date: ______________________________ Page: ____________ Of ____________

Comment Number	Section/ Paragraph	Page Number	Comment

Software Reengineering Assessment Handbook, Version 3.0. 1/97

APPENDIX E

REENGINEERING SIZE ADJUSTMENT (RESIZE) MODEL

This appendix describes how the REengineering SIZE Adjustment (RESIZE) model could be used to estimate equivalent new source lines of code for reengineering efforts. It was developed and provided by:

Mr. John Clark
Comptek Federal Systems, Inc.
2877 Guardian Lane, Suite 200
Virginia Beach, VA 23452
(804) 463-8500 x 316 (Voice)
(804) 463-5675 (Fax)
clark@comptek.com

Mr. Mike Wood
3802 Woodlawn Ave North
Seattle, WA 98103
(206) 547-7853 (Voice)

E.1 Introduction

Parametric cost models focus on new software development, and nearly all of them utilize Source Lines of Code (SLOC) or Delivered Source Instructions (DSI)[1] as the major cost driver (SLOC and DSI are essentially equivalent terms). Since no parametric cost model is currently available specifically for software reengineering, a prototype method is provided in this appendix to adapt existing software estimating models to estimate reengineering cost by adjusting size. The methodology is based on an enhancement of a previous work: *A Reengineering Economics Model* [Wood 92], and is an extension of the concepts of software adaptation [Boehm 81, page 134] and reuse [Londeix 87, page 128]. This appendix was first drafted as part of the *Reengineering Economics Handbook (REH)* in the *Proceedings* of the First Software Reengineering Workshop [JLC 92].

Reengineering may be viewed as similar to adaptation and reuse in that it involves the modification and/or reuse of existing code and associated documentation. The assumption is made that modified and reused code/documentation requires less effort to develop than new code/documentation and results in a reduction of size. Since existing cost models focus on new development, in adaptation/reuse we compute an "equivalent" number of new lines of code (i.e., size adjustment) for input to the cost model. For a reengineering project, a proposed replacement for Boehm's formula for adaptation and Londeix's formula for reuse is provided herein to account for the different processes of each reengineering strategy. This method applies to the more complex reengineering strategies (restructure, translate, redocument, etc.). Costs for simpler strategies (e.g., reformatting or other automated methods which require minimal effort such as running existing code through an automated tool as a batch job) may be estimated using other methods.

This specific method applies to parametric cost models, which are based on new lines of code and utilize an adaptation formula for size adjustment.

A list of acronyms used in this appendix is provided at the end of the appendix.

---

[1]Most models that utilize function points convert them internally to lines of code. Only one, CHECKPOINT, is known to use function points as its internal measure and takes SLOC and converts them to function points and vice versa. Function point-based models are more evolved or calibrated for MIS applications. SLOC remains the standard for embedded systems, but function points are gaining acceptance.

Preliminary design analyses, limited code translation, and benchmark testing are recommended, as required, to estimate the reengineering factors defined in this method. The method is complex. Software size and cost estimation require a high level of experience and expertise. As with any model, each organization must build its own historical database.

E.2 Adaptation/Reuse

In order to use existing new development cost models to predict adapted (reused) software cost, the software size is estimated in terms of equivalent new lines of code. This technique is referred to as size adjustment. Figure E-1 illustrates existing software to be adapted plus new added software. In this example, some of the existing software shown is not reusable and must be redeveloped. The remainder of the existing software can be adapted, resulting in an equivalent number of new lines of code with less size and effort than for new software.

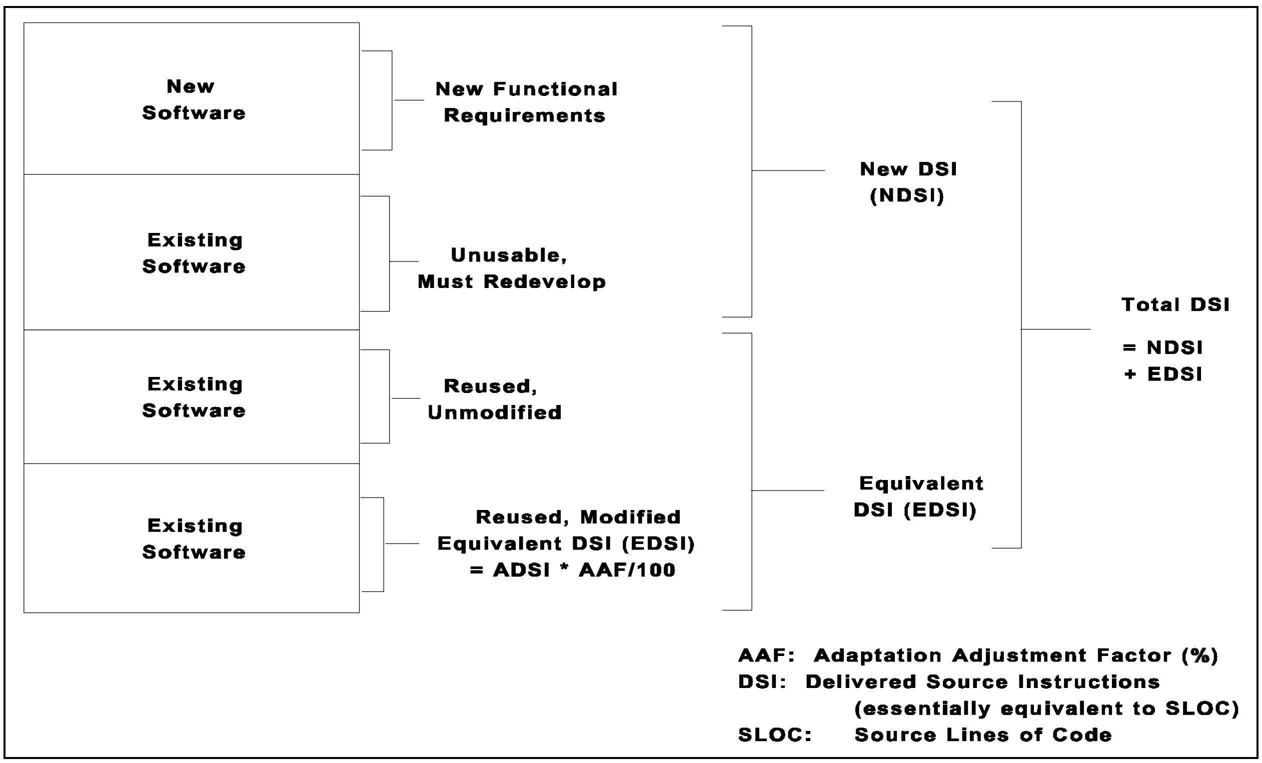


Figure E-1. Adaption/Reuse

Boehm recommends applying three adaptation factors (Table E-1) to estimate the equivalent number of new lines of code. [Boehm 81, page 134]

Factor (Percent)	Coefficients of Factor for Average Projects	Coefficients of Factor for Small Embedded Mode Projects
Percent Design Modified (DM)	0.40	0.40
Percent Code Modified (CM)	0.30	0.40
Percent of Integration Required for Modified Code (IM)	0.30	0.20
Total	1.00	1.00

Table E-1. Effort Distribution by Development Phase

DM includes only high-level and detailed design. CM includes only code and Computer Software Unit (CSU) test. IM includes only Computer Software Component (CSC) testing and integration at the Computer Software Configuration Item (CSCI) level. Actual effort distributions may vary from project to project and depend on the product size and development mode. [Boehm 81, page 137] Effort distributions may also vary depending on the development language (e.g., Ada projects are more front-loaded than FORTRAN projects), development, methodology, experience level, and productivity of the organization.

An Adaptation Adjustment Factor (AAF) in percent is calculated and applied to the total Adapted Deliverable Source Instructions (ADSI) to calculate the Equivalent DSI (EDSI). The coefficients in Table E-1 are used depending on the type of project.

$$AAF = 0.40(DM) + 0.30(CM) + 0.30(IM) \quad (1)$$

(for average projects)

or

$$AAF = 0.40(DM) + 0.40(CM) + 0.20(IM) \quad (2)$$

(for small embedded mode projects)

$$(3) \qquad EDSI = ADSI * \frac{AAF}{100}$$

For example, if 10,000 lines of code are to be adapted for an average project and DM = 30%, CM = 70%, and IM = 90%, then adapting/reusing original code results in only 6,000 "equivalent" new lines of code, i.e.,

$$(4) \qquad AAF = 0.40(30\%) + 0.30(70\%) + 0.30(90\%) = 60\%$$

$$(5) \qquad EDSI = 10{,}000 * \frac{60\%}{100} = 6{,}000$$

and

Note that if DM = CM = IM = 100%, then AAF = 100% and EDSI = ADSI; i.e., no reduction in size occurs which results in a 100% new effort since nothing is reused.

E.3 Reengineering

A limitation of the above adaptation method is that the cost analyst must provide adaptation estimates for the three development phases (corresponding to DM, CM, and IM) at a very high level of granularity. DM, CM, and IM can be estimated for reengineering since the procedure is similar to adaptation; however, the high level of granularity remains. (This method is discussed later in this appendix.) A more detailed method of determining the equivalent software size for adapted/reused software is described by B. Londeix in *Cost Estimation for Software Development.* [Londeix 87, page 128] The process described below for reengineering is based on a modification of the Londeix method.

Londeix breaks the project phases into sub phases, determines the equivalent lines of code for each sub phase, and computes the total equivalent lines of code. High level specification (i.e., requirements) and high level testing (i.e., system testing) are included. The method described in this appendix modifies Londeix's method by providing formulas for equivalent lines of code for each of the development sub phases based on reengineering size factors.

The reengineering estimate depends on the degree to which each of the design entities (objects, modules, components, etc.) in the original application may be adapted/reused. To evaluate the effort to adapt/reuse/reengineer existing code and/or documentation, the distribution of effort by phase for new development projects (all new software) must be examined in greater detail than for DM, CM, and IM. For example, the percentages shown in Table E-2 may be used to allocate effort by sub phase (task) during a new development project. Note that the effort distribution in Table E-2 is an example and is representative of an average project per equation (1). Actual effort distribution may vary from project to project and depends on the product size, development mode, language and methodology employed, and the experience level and productivity of the organization. Each organization should refine this technique by building its own historical database and its own distribution as recommended by Boehm. Also note in development effort percentages (corresponding to DM, CM, and IM) do not include the System or Software Requirements Analysis (or earlier) phases or the Development Test & Evaluation (DT&E) (or later) phases. Thus, as shown in Table E-2, additional effort must be added for these added phases depending on the cost model employed. Note also the final reengineering size estimate is not very sensitive to small changes of the distributions shown in Table E-2.

Factor	Description	Development Effort % *	Additional Effort % **
RA	** Requirements Analysis	0	9
	* Design		
HA	High-level Design Analysis	11	
DA	Detailed Design Analysis	20	
	* Code & Unit Test		
CP	Code Preparation	10	
CR	Code Review and Walk-through	2	
UT	Unit Test	15	
	* Integration Test		
IT	Integration Test & Debug	22	
IA	Integration Test Result Analysis	3	
DT	** Development Test & Evaluation (DT&E)	0	19
RD	** Requirements Documentation	0	3
HD	* High-level Design Documentation	3	
DD	* Detailed Design Documentation	6	
UD	* Unit Test Documentation (Dev. Folders)	3	
ID	* Integration Test Documentation	5	
TD	** DT&E Test Documentation	0	3
	Total	100%	34%

* Items included in the new software development phases totaling 100% of the software development period (design, code, unit test, software integration & test).

** Approximate additional effort not included in the new software development phases

Table E-2. Effort Distribution by New Development Sub-Phase (for average projects)

For reengineering, the new development effort distribution in Table E-2 is modified for reengineering projects. Each sub phase (task) falls into one of three categories:

- Tasks which must always be fully completed for reengineering.
- Tasks which do not need to be completed for reengineering.
- Tasks whose scope depends on the reengineering size factors.

E.3.1 Reengineering Size Factors

In reengineering, the analyst is required to estimate three factors (Table E-3) which contribute to the equivalent software size. These factors are tied to the sub phases of a reengineering project.

Factor	Description	Sub-Phase
REHD	High level Design Reuse	High level Design Analysis
RELD	Low level Design Reuse/Code Salvage	Detailed Design Analysis
TRAN	Translation Efficiency	Code Preparation

Table E-3. Reengineering Size Factors

High level Design Reuse. This metric indicates the degree to which the existing high level design can be recovered (reused). REHD is an effort factor that varies from 0.0 (no reuse) to 1.0 (full reuse).

To estimate REHD, the analyst may create an inventory of high-level design entities (modules, components, etc.) of the original application and the proposed application. Techniques to estimate REHD may vary and have different degrees of granularity and accuracy (e.g., an inventory of CSCs or an inventory of CSUs). REHD is the ratio of surviving high level design entities to total entities.

Low-level Design Reuse. This metric represents the degree to which the existing low-level design and/or code can be recovered (reused/salvaged) and depends directly on the quality of the original software. (Quality measures should be identified within the reengineering process so that the existing low-level design and/or code can be analyzed consistently.) RELD is an effort factor that varies from 0.0 (no reuse) to 1.0 (full reuse).

To estimate RELD, the analyst may conduct several benchmark conversions of representative low-level design entities and/or code (procedures, tasks, subroutines, units, etc.) to observe the differences between the original and the redesigned entities and/or code. Techniques to estimate RELD may vary and have different degrees of accuracy (e.g., a count of lines of code reused, a count of paragraphs reused of a detailed design document, or a weighted average percent of CSUs. Each technique would be categorized by None (0%)/Low (25%)/ Medium (50%)/High (75%)/Full (100%) reuse. RELD is the ratio of surviving low level entities and/or code to total low level entities and/or code.

Translation Efficiency. This metric indicates the efficiency of the source code translation process. TRAN is an effort factor that varies from 0.0 (no translation) to 1.0 (full translation). Values for TRAN represent the work required to translate from the existing language to a new language. If the existing and target languages are the same, this number is 1.0. A typical value for manual translation might be 0.5, indicating that manual translation takes half the effort of writing new code. Some automatic translators are advertised to be 85-95% efficient, though the effort required to code the remaining 5-15% may be disproportionately large.

Automatic translators do not handle idiosyncrasies of the source language nor do they translate code using much, if any, of the advantageous features of the target language. If this reengineering strategy does not have prototypes and benchmarks that apply to a given project, it is very risky and should be addressed as very inefficient. This should be factored into the translation efficiency estimate.

To estimate TRAN, the analyst may conduct benchmarks to translate several representative units or lines of code. TRAN is the ratio of translated units or code to total units or code, or the proportion of time saved by translating existing code versus writing new code.

Document Production and Code Review/Walk-through. In general, for complex reengineering strategies, most of the document production and code review/walk-through tasks must be fully completed regardless of the amount of design or code reuse. This is necessary to conform to the documentation standards dictated by the current project plan and the need to understand the code. In some reengineering situations, the documentation may be out-of-date or non existent and must be created or the code may have been developed to poor coding standards and patched extensively. The amount of analysis required to produce the information in the documents and the effort required to produce the code will vary depending on the condition of the application and the availability of translation tools. Londeix assumes a smaller effort for redocumentation/update of unchanged reused documentation as compared to new documentation. In this modification of Londeix's method, two options are provided: full and reduced documentation. Londeix also assumes a smaller effort for code review/walk-through of unchanged reused code. Two options are provided herein: full and reduced code review/walk-through.

Values for REHD, RELD, and TRAN will depend on the reengineering strategy employed. Table E-4 includes possible ranges of values for some of the reengineering strategies. Typical ranges could include: Medium 0.4 - 0.6, High 0.6 - 0.8, Very High 0.8 - 1.0.

Strategy	REHD	RELD	TRAN
Translation	Very High	High	Tool Dependent
Restructuring	High	Medium	*
Reengineering for Reuse	Medium	Medium	*
* Very high if the target language equals the source language; tool dependent otherwise.			

Table E-4. Possible Values for Reengineering Size Factors

E.3.2 Reengineering Size Adjustment for Each Sub Phase

Modification of the new development effort in Table E-2 for application to reengineering projects varies by phase. Effort during the requirements phase is constant (does not depend on the level of design restructuring) while effort during the design phases depends on the level of design restructuring required. Coding effort depends on the level of design restructuring and the efficiency of code translation. Effort during the test phases is, in turn, dependent on design and code changes.

Requirements Analysis and Documentation. Londeix assumes full high-level specification/design effort for all of the software as if it were all new. However, in this handbook, this is interpreted as Requirements Analysis (RA) and Requirements Documentation (RD), and none of this effort is included in the resizing estimate because RA and RD represent efforts not included in the development phases corresponding to AAF (Table E-2). Additional effort may be added. In theory, for reengineered code, the software requirements are embodied in the original implementation. In general, while most of the requirements analysis (RA) may be eliminated, a full requirements document (RD) must usually be prepared based on the current system behavior and to conform to current documentation standards.

High-level Design. At a minimum, high level design documentation (HD) must be prepared. High level design analysis $(HA)_R$ effort depends on the amount of design restructuring required at the program level (i.e., the high level design analysis to replace the non reusable high level design).

$$(6) \qquad (HA)_R = (HA) * [1REHD]$$

Detailed Design. Detailed Design Documentation (DD) must always be prepared. Detailed Design Analysis $(DA)_R$ effort depends on the portion of the original detailed design that can be reused/salvaged from the existing software (i.e., the detailed design analysis to replace the non-reusable low level detailed design).

$$(7) \qquad (DA)_R = (DA) * [1RELD]$$

Code. Code review and walk-through (CR) must be conducted for all units even if little or no code is actually changed. Code preparation $(CP)_R$ effort is the sum of the effort required to write replacement code plus, if applicable, the effort to translate original code. Total replacement code is estimated as being proportional to the level of redesign (high level and detailed design) required. Translation effort depends on the design recovered and the efficiency of the translation process. Let the Design Analysis Ratio (DAR) represent the design which is non recoverable and requires effort to redesign (i.e., the non reusable high level Design Analysis plus the non reusable Detailed Design Analysis, divided by the total high level and Detailed Design Analysis).

$$(8) \quad DAR = \frac{HA_R + DA_R}{HA + DA} = \frac{(HA)[1REHD] + (DA)[1RELD]}{HA + DA}$$

Code preparation is the effort to recode the non reusable high level and non reusable detailed design, plus the reusable high level and reusable detailed design that did not translate.

$$(9) \qquad (CP)_R = (CP)\,[\,DAR + (1DAR)\,(1TRAN)\,]$$

or, the code preparation effort to recode what did not translate, plus what did translate but required redesign.

$$(10) \qquad (CP)_R = (CP)\,[(1TRAN) + (TRAN)\,DAR\,]$$

Unit Test. Unit Test Documentation $(UD)_R$ must always be fully prepared and reviewed. Unit Test and Debug Effort $(UT)_R$ is proportional to the coding effort $(CP)_R$ (i.e., the unit test effort to retest the code in equation 10).

$$(11) \qquad (UT)_R = (UT)\,[\,(CP)_R \;/\; (CP)\,]$$

or

$$(12) \qquad (UT)_R = (UT)\,[\,(1TRAN) + (TRAN)\,DAR\,]$$

Unit Regression Testing. As advocated by Londeix, regression testing and validation are limited to the unchanged portions of changed entities (CSCs, procedures, modules, etc.). However, in this handbook, this unit regression testing effort is already included in equation (12) above based on the inventory (count) of whole (vice partial) entities to compute REHD and RELD used in the calculation of DAR.

Integration and Test. Integration Test Case Documentation $(ID)_R$ must always be prepared and reviewed, even if the original system contains a working prototype. (Substantial savings are realized during integration by having a working prototype in the original system.) Integration Test Results Analysis $(IA)_R$ can be performed by observing the behavior of the working system.

Integration testing is proportionally more sensitive to design changes than to unit test. Assuming that thorough unit testing has been completed, most errors discovered during integration are design errors. Integration Test and Debug Effort $(IT)_R$ and the corresponding analysis effort $(IA)_R$ are, therefore, proportional to the Design Analysis Ratio. The effort to integrate and test the redesign is

$$(13) \qquad (IT)_R = (IT) * DAR$$

and the effort to analyze the integration and test results is

$$(14) \qquad (IA)_R = (IA) * DAR$$

Integration Regression Testing. Londeix advocates full regression testing and validation of all of the unchanged reused code. However, in this handbook, regression testing and validation are limited to the unchanged portions of the changed entities. This effort is already included in equations (13) and (14) above based on the inventory (count) of whole (vice partial) entities to compute REHD and RELD used in the calculation of DAR. While testing of the entities can be limited to the unchanged reused code, verification of the entire system's integrity dictates that all code be regression tested and validated. Additional regression testing effort may be added. Full regression testing, validation, and analysis would result by setting DAR=1.

Development Test and Evaluation (DT&E) and Documentation. Londeix advocates full high-level testing effort for all of the software as if it were new. However, in this handbook, this effort is interpreted as Development Test and Evaluation (DT&E), i.e., System Test, and is not included in the resizing estimate because this effort is not included in the development phases corresponding to AAF (Table E-2). Additional effort for DT&E may be added.

E.3.3 Reengineering Size Adjustment Total

In reengineering, as in the formula for adaptation, a Reengineering Adjustment Factor (RAF) may be calculated and applied to the total Reengineered SLOC (RSLOC) to calculate the Equivalent SLOC (ESLOC). (Hereafter SLOC is used instead of DSI.)

$$(15) \qquad ESLOC = (RSLOC) * \frac{RAF}{100}$$

where RAF is the sum of all of the factors in Table E-2 as modified by Equations (6) through (14).

RAF = All documentation and code review/walk-through effort plus

$$(16) \quad (HA)_R + (DA)_R + (CP)_R + (UT)_R + (IT)_R + (IA)_R$$

thus

(17) $RAF = HD + (HA)\,[1\text{-}REHD] + DD + (DA)\,[1\text{-}RELD]$
$+ CR + (CP)\,[\,(1\text{-}TRAN) + (TRAN)\,DAR\,]$
$+ UD + (UT)\,[\,(1\text{-}TRAN) + (TRAN)\,DAR\,]$

$$+ ID + (IT + IA)\ DAR$$

Equation (17) is based on the stated requirement for fully completed documentation and full code review/walk-through. If the original unmodified documentation is adequate and can be reused as is, and full code review/walk-through is not required, then equation (17) may be rewritten as follows:

$$\begin{aligned}(18)\quad RAF \;=\; & (HD + HA)\,(1\text{-}REHD) + (DD{+}DA)\,(1\text{-}RELD)\\ & + (CR + CP)\,[\,(1\text{-}TRAN) + (TRAN)\,DAR\,]\\ & + (UD + UT)\,[\,(1\text{-}TRAN) + (TRAN)\,DAR\,]\\ & + (ID + IT + IA)\,DAR\end{aligned}$$

In equation (18), only the directly affected documentation and code review/walk-through are included in the effort. On the other hand, if only full documentation is needed, then, from Table E-2:

$$(19)\quad RAF \;=\; HD + DD + UD + ID \;=\; 17\%$$

Note that portions of Equation (17) may be mapped to DM, CM, and IM as follows:

$$(20)\quad 0.40(DM) = HD + (HA)\,[1\text{-}REHD] \;+\; DD + (DA)\,[1\text{-}RELD]$$

$$\begin{aligned}(21)\quad 0.30(CM) = & \; CR + (CP)\,[\,(1\text{-}TRAN) + (TRAN)\,DAR\,]\\ & + UD + (UT)\,[\,(1\text{-}TRAN) + (TRAN)\,DAR]\end{aligned}$$

$$(22)\quad 0.30(IM) = ID + (IT{+}\,IA)\,DAR$$

Finally, if it is desired to use DM, CM, and IM in an average reengineering situation, including redocumentation, then an alternative form of AAF may be written as follows (refer to Table E-2):

$$(23)\quad AAF = 0.31(DM) + 0.27(CM) + 0.25(IM) + 0.17(ND)$$

where ND is the percent of new plus modified documentation pages. As noted in the beginning of this section, actual effort distribution in Table E-2 and therefore in all of the above equations may vary from project to project and depends on a number of factors.

E.4 Case Study Example

The case study used for this example has 31,574 lines of CMS-2 code to be reengineered. The average project effort distribution, as shown in Table E-2, is assumed. Three reengineering strategies were indicated (redocument, translate, and restructure).

Redocumentation. It was assumed that all new documentation must be produced. Equation (17) then becomes:

$$(24)\quad RAF = HD + DD + UD + ID = 17\%$$

and the equivalent code size is

$$(25)\quad ESLOC = (SLOC) * \frac{RAF}{100} = \frac{31{,}574 \text{ x } 17\%}{100} = 5{,}368$$

Translation. Manual code translation is assumed, resulting in TRAN = 0.50. Most of the high level and low level design is retained. Preliminary design analysis indicates that 9 of 10 CSCs can be reused, resulting in REHD = 0.90. Due to the desire to take advantage of the Ada constructs, a weighted average of CSUs indicates that only 70% of the low level design/code can be reused, resulting in RELD = 0.70. Full redocumentation and full code review/walk-through are required. Equation (17) then becomes:

$$\begin{aligned}(26)\quad RAF = \; & HD + (HA)\,[\,1\text{-}REHD\,] + DD + (DA)\,[\,1\text{-}RELD\,] \\ & + CR + (CP)\,[\,(1\text{-}TRAN) + (TRAN)\,DAR\,] \\ & + UD + (UT)\,[\,(1\text{-}TRAN) + (TRAN)\,DAR\,] \\ & + ID + (IT + IA)\,DAR\end{aligned}$$

$$(27)\quad RAF = 3 + (11)\,[\,1- 0.90] + 6 + (20)\,[\,1- 0.70\,]$$
$$+ 2 + (10)\,[\,(1- 0.50) + (0.50) * 0.23\,]$$
$$+ 3 + (15)\,[\,(1-0.50) + (0.50) * 0.23\,]$$
$$+ 5 + (22 + 3)\,0.23$$

$$RAF = 47\ \%$$

$$(28)\quad ESLOC = 31{,}574 \times 0.47 = 14{,}840$$

Restructuring. Moderate restructuring is required for the high level design, but significant restructuring is required for the low level design/code. Preliminary design analysis indicates that only 7 out of 10 CSCs can be reused, resulting in REHD = 0.70, but a weighted average of CSUs indicates that only 50% of the low level design/code can be reused, resulting in RELD = 0.50. Since no translation is required, TRAN = 1.0. Redocumentation is required only for the changed areas. Equation (18) then becomes:

$$(29)\quad RAF = (HD + HA)\,(1\text{-}REHD) + (DD+DA)\,(1\text{-}RELD)$$
$$+ (CR + CP)\,[\,(1\text{-}TRAN) + (TRAN)\,DAR\,]$$
$$+ (UD + UT)\,[\,(1\text{-}TRAN) + (TRAN)\,DAR\,]$$
$$+ (ID + IT + IA)\,DAR$$

$$(30)\quad RAF = (3 + 11)\,(\,1 - 0.70) + (6 + 20)\,(\,1 - 0.50)$$
$$+ (2 + 10)\,[\,(\,1 - 1) + 1 \times 0.43\,]$$
$$+ (3 + 15)\,[\,(\,1 - 1) + 1 \times 0.43\,]$$
$$+ (5 + 22 + 3) \times 0.43$$

$$RAF = 43\ \%$$

$$ESLOC = 31{,}574 \times 0.43 = 13{,}577$$

E.5 Summary

A candidate method is provided herein to adapt new development parametric cost models to estimate the cost of reengineering. The user of this method is encouraged to provide comments and suggestions. Quantitative results are especially critical as efforts are made to improve this technique.

E.6 List of Acronyms

AAF	Adaptation Adjustment Factor
ADSI	Adapted DSI
CM	Percent of Code Modified
CP	Code Preparation
CR	Code Review and Walk-through
CSCs	Computer Software Components
CSUs	Computer Software Units
DA	Detailed Design Analysis
DAR	Design Analysis Ratio
DD	Detailed Design Documentation
DM	Percent of Design Modified
DSI	Delivered Source Instructions
DT/DT&E	Development Test and Evaluation
EDSI	Equivalent DSI
ESLOC	Equivalent SLOC
HA	High Level Design Analysis
HD	High Level Design Documentation
IA	Integration Test Result Analysis
ID	Integration Test Documentation
IM	Percent of Integration Required for Modified Code
IT	Integration Test and Debug
NDSI	New DSI
RA	Requirements Analysis

RAF	Reengineering Adjustment Factor
RD	Requirements Documentation
REH	Reengineering Economics Handbook
REHD	High Level Design Reuse
RELD	Low Level Design Reuse/Code Salvage
RSLOC	Reengineered SLOC
SLOC	Source Lines of Code
TD	DT&E Test Documentation
TRAN	Translation Efficiency
UD	Unit Test Documentation (Development Folders)
UT	Unit Test

APPENDIX F

CHECKPOINT® SOFTWARE ESTIMATING MODEL

This appendix describes how CHECKPOINT® can be applied to reengineering efforts. It was prepared by Mr. Capers Jones of Software Productivity Research and edited by Mr. John Clark of Comptek Federal Systems.

Mr. Capers Jones
Software Productivity Research, Inc.
1 New England Executive Park
Burlington, MA 01803-5005
(617) 273-0140 (Voice)
(617) 273-5176 (Fax)
capers@xanadu.spr.com

F.1 Introduction

CHECKPOINT supports assessment of software cost and schedules. It predicts the cost of source code and major deliverables and also measures important software strengths and weaknesses. CHECKPOINT provides continuous monitoring of project status through completion, including "what-if" capabilities for evaluating alternative options. Measured data can be automatically normalized to project size in terms of function points, feature points, or lines of code. CHECKPOINT uses a knowledge-based evaluation format that ranks project variables against an industry scale. Variables have been developed from studies at hundreds of major corporations and government facilities. Measurement criteria cover "hard" data such as staffing levels, schedules, costs, project size and number of defects, and also "soft" data such as skill levels, environmental factors, and other expectations or constraints established by management and clients.

Application Types: Commercial, Real-time/reactive, Scientific/engineering.
Tasks: Cost and size estimation, Metrics collection/analysis.
Methods/Notations: Function points.
Hardware Platforms: HP9000, IBM-PC (compatibles), Motorola 8000, Sun SPARC.
Operating Systems: MS-DOS, UNIX.

F.2 Model Inputs and Sample Case Study Description.

Model Inputs:

The CHECKPOINT tool operates in the following sequence when performing an estimate:

Users provide basic information about the project to be estimated such as:
- Whether it is military or civilian
- Whether it is new work or an enhancement
- Whether or not DoD standards will be followed
- Whether it is systems, embedded, MIS, or hybrid software
- Whether overtime is paid, unpaid, or a mixture.

This information is entered by using pull-down menus and standard check lists. No free-form data entry is required. Since CHECKPOINT includes a fully automated assessment questionnaire of about 100 questions, and can also be used for task-level data collections and quality measurement, the entire set of CHECKPOINT inputs is too large for this report. The entire CHECKPOINT questionnaire in printed form is about 50 pages in length.

CHECKPOINT then selects similar projects from its integral knowledge base to use as the starting point for the estimation process. Further user-supplied information will refine the estimate.

Users may then input size in LOC or function point form, or they can use the integral CHECKPOINT sizing function.

If users invoke the CHECKPOINT sizing function, data must be entered to either estimate function points or approximate function points. (The approximation feature fills in missing data from similar projects.) Once the function point total of the application is known, CHECKPOINT can predict source code size, and also predict document sizes for more than 50 kinds of specification and planning documents. CHECKPOINT also includes full bi-directional conversion logic between LOC metrics and function points. (Note: for this case study the 28.4 KLOC of CMS-2 is equivalent to 270 function points. The 3.1 KLOC of assembly language is equivalent to 15 function points, so the entire case study equals 285 function points. Such data is provided automatically by CHECKPOINT's conversion feature.)

Users then answer some or all multiple-choice questions about tools, processes, experience levels, quality control approaches, and other factors known to influence software projects. (Questions that are not answered are excluded from consideration and do not affect the outcome of the estimate.) More than 100 influential factors are included in CHECKPOINT. An example of a typical CHECKPOINT input question format follows. This example is relevant to the case studies illustrated herein:

Functional Novelty?

(1) Conversion or functional repeat of a well known program
(2) Functional repeat, but some new features
(3) Even mixture of new and repeated features
(4) Novel program, but with some well understood features
(5) Novel program, of a type never before attempted

Although not requested by the case study, it should be noted that CHECKPOINT also includes a complete and automated software process assessment capability. Estimating templates are available for all five SEI CMM levels.

At any point, users may VIEW the current estimate to see the results of all answers to input to date. Users may select various levels of granularity when using the VIEW function. The supported levels include:

Project level data (no inner structure displayed)
Phase level data (8 standard phases displayed)
Activity level data (25 standard activities displayed)
Task level data (200 standard tasks displayed)

When used at the project or phase level, CHECKPOINT simulates the appearance of many macro-estimation tools. However, CHECKPOINT is actually a micro-estimator whose level of estimating granularity is far greater than any current macro estimation tool. The activity and task level estimates illustrate CHECKPOINT's micro-estimation capabilities.

When a user is satisfied that all relevant information has been given, the estimate can be stored or directed to a printer.

Once an estimate is stored, users can modify the estimate such as changing assumptions about languages, tools, or processes. The revised estimate can then be compared against the prior scenario using a standard side-by-side mode that shows both versions with all differences highlighted.

CHECKPOINT also has a measurement mode for recording historical data. As a project proceeds, actual information can supplant the estimated data until the information consists entirely of historical data. At any point, CHECKPOINT will estimate time to complete, cost to complete, or any missing item such as the size of a deliverable that the user does not have as historical data.

CHECKPOINT's maintenance and enhancement estimates default to a 5-year period, but users may select any period up to a maximum of 20 years.

CHECKPOINT also includes quality and reliability estimation and will estimate defect potentials and defect removal efficiency levels. CHECKPOINT will also estimate field-reported defects by severity level for up to 20 years.

For maintenance and enhancement estimates, CHECKPOINT will estimate the gradual increase in complexity over time (as measured by cyclomatic complexity). CHECKPOINT also includes inflation rate tables, to adjust cost and compensation levels for long-range estimates. Users can override the default assumptions and set inflation rates to match local conditions.

Case Study:

The basic case study is for 31,574 SLOC written in CMS-2 (90%) and Assembly (10%). This case study is roughly equal to 285 function points. The application is more than seven years old.

Scenarios requested were:

Status quo of software (continued maintenance)
Redocumentation of software
Restructure of software
- Automated restructure

- Manual restructure
Translation of code into Ada
Retargeting to a new system and configuration
Data reengineering
Reverse engineering without redevelopment
Reverse engineering followed by redevelopment
Redevelopment
Retirement

Note: Other possibilities exist that were not stated in the scenarios. One of these is error-prone module removal. Another is to explore the results of various SEI CMM levels on the redevelopment scenarios.

Civilian and military software projects differ in a number of important respects. When CHECKPOINT is set to estimate a military project, it makes the following assumptions:

- Relevant military standards will be followed.
- Paperwork is the dominant military software cost element, and all planning and specification documents are larger than civilian norms.
- Defect removal activities will include critical design reviews (CDR) independent verification and validation (IV&V), and other activities that are essentially unique to the military environment.

Function points remain constant regardless of the programming language used. This is one of their key virtues. Function points are now the most widely used software metric in 18 countries, including the U.S. The military has lagged in modern functional metrics, but the Air Force is now starting to mandate function points.

F.3 Status Quo

The starting year of the project was set to 1985, and the project was estimated using a general template more or less equivalent to SEI CMM level 1 since the majority of all companies are at that level today, and even more were at that level in 1985. Some of the results of attempting replicate the original development cycle were: 89.84 months of paid time; costs of $907,520, and a schedule of 19.75 calendar months resulting in a nominal delivery date of March 1987 for the first release. Productivity rates were 351 LOC per man month or 3.17 function points per man month which are relatively average for SEI level 1 projects of this size range. (Note: the dates assumed were those used in the March 1995 Version 2.0 of this handbook, and were not updated to reflect the current date.)

Since the other scenarios are driven by the economics of continued maintenance for 12 years, CHECKPOINT's estimate of inflated annual maintenance costs from 1994 forward are relevant to the other choices. Results are summarized in Table F-1.

CHECKPOINT operates in both macro-estimation and micro-estimation modes, and also can be used in measurement mode. In addition, CHECKPOINT can be used to automate the software process assessment concept and includes a full assessment questionnaire using the Software Productivity Research (SPR) assessment approach.

The CHECKPOINT tool can also sort the assessment responses into sets where the answers are better than U.S. norms, at U.S. norm levels, or below U.S. norms.

The complete set of output reports for a single project can total to more than 150 pages when CHECKPOINT is used for both as a micro estimator and to collect assessment data. In order to meet the space limitations, the sample pages are not shown.

The total number of screens in the full CHECKPOINT product totals to more than 200. This is because the CHECKPOINT tool integrates the functions of:

Software process assessments
Full sizing of all deliverables
Metric conversion between LOC and function point metrics
Macro cost and schedule estimation
Micro cost and schedule estimation
Maintenance and enhancement estimation for up to 20 years
Quality and reliability estimation
Risk and value analysis
Side by side comparisons of alternate scenarios
Measurements
Portfolio analysis and statistical aggregation of data
Importing and exporting of data to off-line tools such as spreadsheets
Choice of graphic or tabular display of data
Tool catalog with more than 500 tools
Programming language catalog supporting more than 400 languages

Due to size and space limitations the screens are not shown.

CHECKPOINT's inflation-rate table can be set to constant dollars, or can be set to any degree of annual inflation from 1% to more than 100%. In this example, a value of 6% is selected. Salary and burden rates can also be set independently. Paid and unpaid overtimes are also adjustable, as are assumptions about overtime premium pay, working hours, vacation days, holidays, and other factors associated with accurate time and cost recording.

In addition, the growth of new functions (and the deletion of dead functions) can also be set. From the ground rules of the case study, this scenario is set for a 7% annual increment of new functionality. (The actual setting was for 9% new functions and 2% removal of obsolete functions, leading to a 7% net annual change.)

The combination of increasing cyclomatic complexity over time, and inflation of about 6% per year, leads to a troubling future if normal routine maintenance is continued indefinitely. In real life as well as with this case study, some form of geriatric treatment appears to be strongly indicated. Projects with high cyclomatic complexity levels tend to destabilize rapidly, and will eventually reach two significant plateaus: the point where complete testing is no longer feasible, and the point where every change has a high probability of causing an error.

F.4 Redocumentation

In real life, a strategy of manual redocumentation has never been shown to be economically effective or viable. CHECKPOINT's estimates for the size and effort associated with redocumentation illustrate why this is not necessarily a good scenario:

Documentation Item	**Estimated page Count**	**Estimated cost**
New requirements	331	$62,570
New functional design	320	$47,390
New logic specification	318	$76,600
New users guide	328	$105,200
New reference manual	276	$90,500
New operators guide	193	$75,400
TOTAL	1,766	$457,660

Note that the author has seen a number of attempts at manual redocumentation but none has succeeded very well. The worst case was a situation where the technical writers threatened to quit the company rather than continue with the project.

Because empirical data does not support this scenario as being economically valid, it was considered unnecessary to include more data.

F.5 Restructuring

Automatic restructuring of aging source code has been a commercial technology since 1985. A number of automatic translation tools are available for CMS-2 which are advertised to include restructuring capabilities, so this scenario includes fully automatic restructuring of the CMS-2 portion.

The empirical data on automated software restructuring indicates three phenomena of significant interest: (1) Maintenance effort is reduced by roughly 40%; (2) Most restructuring tools increase the volume of source code by about 10%; (3) Cyclomatic complexity levels are reduced to safe levels.

The CHECKPOINT tool allows users to specify any restructuring interval, from "none at all" to "annual restructuring." In this scenario, restructuring was specified to be carried out at four year intervals. Once the cyclomatic complexity is reduced, it is not necessary to restructure more frequently because the entropy will not increase very rapidly. Results are summarized in Table F-1.

Comparing the results of the automated restructuring scenario with the 12-year maintenance costs of the unstructured product (Status Quo) illustrates some very attractive economic advantages: a reduction of about 2 to 1 in cumulative maintenance costs.

Automatic restructuring itself is so cheap and quick that it is irrelevant to show it. Assuming that the CMS-2 restructuring tools are as rapid and effective as civilian tools such as RECODER or STRUCTURED RETROFIT, the entire effort of automatically restructuring this application would take less than one hour of human effort and therefore cost less than $100. Of course the costs of the restructuring tools themselves can exceed $100,000, but these costs are normally amortized across multiple projects and not assigned to a single application. (Especially not to a fairly small application such as this case study.) Retesting is normally not required after restructuring since the process is quite benign if the tools are similar to those mentioned. A test suite can be run afterwards, but costs are minimal since the process adds no bugs.

Although the case study requested that manual restructuring also be included, the empirical data on manual restructuring indicates that it is not an economically viable alternative. What would be a useful alternative was not requested by the case study at all: the removal of error-prone modules. In all large systems and many mid-sized applications, errors are not randomly distributed but clump in a small number of very buggy modules. A general rule of thumb is that 5% of the modules in any application will receive 50% of all post-release bug reports. Manual restructuring of error-prone modules can be cost-effective, although automated restructuring is far superior.

F.6 Translation

This strategy appears to offer some attractive economic advantages. The Ada83 language is substantially more powerful and better structured than either CMS-2 or assembly language. The volume of source code required in Ada was sized by CHECKPOINT as 16.5 KLOC for the same 285 function point application.

The development schedule was 9.23 months, development effort was 14.81 months, and development costs were $114,960. (Pure coding was only $40,760.) Development productivity was estimated to be 26.5 function points per month or about 1,650 net Ada LOC per month.

Future maintenance for the 12-year duration of the Ada version was only about one sixth as expensive as continued maintenance of the existing CMS-2 and assembly language version. The 12-year maintenance costs for the recoded Ada version are quite favorable when compared to the results of Status Quo: indeed the total maintenance over the same period amounts to only about 17% of the Status Quo scenario. Results are summarized in Table F-1.

Because the conversion to Ada83 is assumed to take place during calendar year 1994, maintenance for the current year is zero. Effective maintenance costs do not start until calendar year 1995. CHECKPOINT can also operate on a fiscal year basis, but the case study did not specify calendar years or fiscal years.

It may seem surprising that the cyclomatic complexity of the application can stay constant for a 12-year period. That attribute is a characteristic of projects whose initial structure is very good. A good or excellent initial structure can dramatically slow down the increase of entropy over time, and hence delay or eliminate the onset of the catastrophic problems that can occur when poorly structured applications near the end of their useful life.

F.7 Retargeting

CHECKPOINT includes a standard estimating capability for dealing with the porting of software from one platform to another. (Note that CHECKPOINT itself has been ported from DOS to UNIX and Windows.) However, porting the case study application to another platform without adding any functionality, restructuring, or taking any corrective actions does not offer any significant economic advantages. Such work would be undertaken primarily because the original platform has become obsolete, or to expand market shares. Since neither of these situations was cited in the case study, it was decided to omit showing the restructuring results.

If this scenario were included, accurate estimation with CHECKPOINT would require user inputs as to whether, (1) code conversion tools are available or (2) the code will be manually recoded. The costs are obviously quite different between situations 1 and 2.

If automated code conversion tools are available, the effort devoted to porting can be as little as 1 week. If such tools are not available, but the CMS-2 operates more or less the same on the new host machine, the porting can take about 1 month. If the CMS-2 language is not supported and the code must be rewritten, then the results would be so close to the translation scenario that doing another version would be irrelevant.

F.8 Data Reengineering

This scenario is not directly supported by CHECKPOINT, and it is unlikely that it can be supported by any competitive estimating tool either. The fundamental problem with attempting to explore data reengineering is the lack of any valid metric for expressing data volumes or data quality levels.

All automated estimating tools require some kind of size dimension, such as LOC or function points. So far as can be determined, there is no standard metric for describing the size of data, its quality levels, or any other tangible aspect.

F.9 Reverse Engineering without Redevelopment

This scenario has no economic advantages and apparently was included in the case study simply for the sake of completeness. However, reverse engineering followed by redevelopment offers very attractive economic advantages. See the next section for a full discussion of relevant topics.

F.10 Reverse Engineering Followed by Redevelopment

The case study is not explicit in what technologies or programming languages would be used in this scenario. There is no real-life advantage in going to the trouble of redeveloping an application unless you are prepared to do it well. Therefore, this scenario assumes the use of the new Ada95 programming language, and it also assumes that the redevelopment work will be done by an organization at SEI CMM Level 3 rather than the SEI CMM Level 1 assumption of the original case. (CHECKPOINT estimating templates are available for all five SEI CMM levels.)

The reverse engineering component assumes the presence of state-of-the-art reverse engineering tool suites, such as those marketed by Bachman or Cadre for example. Pure reverse engineering using tools such as this is comparatively inexpensive and would take only a matter of hours or a few days.

This scenario assumes that because of the new Air Force criterion that SEI Level 3 contractors be relieved of some of the burdensome documentation requirements demanded by DoD 2167A, the redevelopment scenario is assumed to follow state-of-the-art civilian practices rather than the more cumbersome and expensive military practices. In particular, the volume of specifications for this scenario is now set to match civilian norms rather than military norms. This means that the volume of specification and planning material is reduced by about 50% compared to normal DoD and military practices.

The new Ada95 language is substantially more powerful than the current Ada83 language. CHECKPOINT's sizing feature estimated a total code volume of only 12,300 Ada95 statements for this entire project, as compared to 31,574 statements in CMS-2/assembly, and 16,500 Ada83 source code statements.

Further, since this scenario assumes SEI CMM Level 3 capabilities, the reuse percent was assumed to be 50%.

Overall, this scenario offers the best economic results of any of the cases. The complete development effort had a schedule of only 7.8 calendar months, paid effort of 14.7 man months, and costs of $147,800. Net productivity was 19.38 function points per man month, or 837 LOC per man month.

It should be recognized that these favorable results are due to a concatenated series of assumptions, including:

Usage of Ada95
SEI CMM 3 assumptions
50% reuse of all major deliverables
Relief from the paperwork burden of current DoD standards

If CHECKPOINT models this scenario using different assumptions, such as reverting to SEI CMM Level 1, or producing the full gamut of normal DoD paperwork, the results would not be so favorable.

The 12-year maintenance costs are also the most favorable with this scenario. (Note that because the redevelopment is assumed to take place during calendar year 1994, maintenance does not commence until 1995.) Results are summarized in Table F-1.

The quality results of this scenario are also excellent. CHECKPOINT's quality estimation facility predicated only 6 latent defects or bugs upon delivery. By contrast, CHECKPOINT's predicted volume of latent bugs at delivery for the Status Quo scenario totaled 55.

As with the prior Ada83 example, achieving an excellent level of structure during development pays off in very low entropy and continued low cyclomatic complexity levels even in the face of repeated changes.

F.11 Redevelopment without Reverse Engineering

This scenario is probably included for logical consistency. Since redevelopment without prior reverse engineering has no economic advantages, the results have not been displayed.

The actual costs of doing this would vary widely based upon the SEI CMM level of the organization, the structure and stability of the current application, and many other factors.

Manual redevelopment does not guarantee a reduction in maintenance costs. In some cases, it has been done so poorly that maintenance even goes up rather than down.

F.12 Retirement

CHECKPOINT does not deal with retirement of software and the withdrawal from service of existing applications. However, in real life this task is sometimes difficult and can even lead to litigation. In several cases, disgruntled clients have sued vendors for attempting to stop supporting current applications that had continuing value to the client's operations.

Table F-1. CHECKPOINT Results for Maintenance Cost and Complexity

	Status Quo		**Restructuring**		**Translation**		**Reverse Then Redevelop**	
Year	**Maintenance Cost**	**Cyclomatic Complexity**	**Maintenance Cost**	**Cyclomatic Complexity**	**Maintenance Cost**	**Cyclomatic Complexity**	**Maintenance Cost**	**Cyclomatic Complexity**
1994	$345,000	18	$207,000	*10	$0	4	$0	2
1995	$393,800	19	$236,280	10	$72,100	4	$18,700	2
1996	$449,700	20	$260,826	11	$81,900	4	$21,300	2
1997	$513,600	20	$292,088	11	$92,900	4	$24,200	2
1998	$586,700	21	$328,552	*10	$105,400	4	$27,400	2
1999	$670,500	22	$375,480	10	$119,600	4	$31,200	2
2000	$766,300	23	$429,128	11	$135,700	4	$35,400	2
2001	$876,100	23	$473,094	11	$154,000	4	$40,700	2
2002	$1,001,900	24	$541,026	*10	$174,700	4	$45,700	2
2003	$1,146,100	25	$618,894	10	$198,200	4	$51,900	2
2004	$1,311,500	26	$681,980	11	$224,900	4	$58,900	2
2005	$1,501,100	27	$780,572	11	$255,300	4	$65,800	2
TOTAL	$9,562,300		$4,640,094	*Restructured	$1,614,700		$421,200	

APPENDIX G

PRICE S SOFTWARE ESTIMATING MODEL

This appendix describes how the PRICE S software estimating model can be applied to reengineering efforts. It was prepared by Mr. James Otte of Lockheed Martin PRICE Systems and edited by Mr. John Clark of Comptek Federal Systems.

Mr. James Otte
Lockheed Martin PRICE Systems
101 Woodman Drive, Suite 12
Dayton, OH 45431
(937) 258-7188 (Voice)
(937) 252-6154 (Fax)
(800) 43 PRICE
james.otte@pricesystems.com

G.1 Introduction

The PRICE Software Model (PRICE S) applies parametric cost-modeling methods for estimating resources for computer software development, along with operations and support. The model is flexible to allow for adapting it to organizational definitions and applications such as reengineering tasks, maturity level measurement, along with deriving software size, and cost and schedule estimates.

G.2 Model Inputs, Sample Case Study Description, and Solution

The PRICE S Model input parameters can be categorized into programmatic, qualitative, and quantitative inputs parameters. Programmatic inputs consist of development and support schedule dates, and economic and accounting input parameters. Qualitative inputs consist of language type, software development personnel experiences, tools/workstations, software specification level, and integration difficulties. Quantitative inputs consist of the functional mix of the software, development and support organization capabilities, amounts of new software and reusable software, software size in terms of number of instructions, and quality level of the delivered software. In addition to the capability of modeling developed software, purchased software and furnished software can be described allowing for total software system integration. Model inputs are described in an Estimating Breakdown Structure (EBS). The EBS is a graphical user interface (GUI) which assists model users in developing software cost hierarchies. The PRICE S EBS also allows model users to evaluate all reengineering strategies within one graphical window.

Case Study

A sample case study was used to demonstrate how PRICE S can be used for accomplishing reengineering assessments. Details and assumptions are described briefly below and during discussions for each strategy. The sample case study will be used to illustrate PRICE S input parameters that should be evaluated or modified for completing an assessment for each of the reengineering candidate strategies listed below:

Maintain Status Quo
Redocumentation
Restructuring
Translation

Retargeting
Data Reengineering
Reverse Engineering
Redevelopment

The sample case study consists of a Management Information System (MIS) developed for the Air Traffic Control (ATC) system. The original software had an expected life of seventeen years, and to date ten years still remain. The software has been fielded for seven years. The original system consisted of 31,574 source instructions. The instructions were written in CMS-2 (28,417 source instructions) and Assembly (3,157 source instructions). The current system is being maintained at a cost exceeding $250K per year. The software will be developed for a Military Ground Specification. Ada was assumed as the higher order language for the translation and redevelopment examples (Note: Other High Order Languages could be used for translation and substituted to the described approach). Since operations and support inputs and cost estimates may be generated from the same input data parameters, a separate paragraph was not devoted to operations and support. The only additional data required for an operations and support estimate is deployment data. The model uses the deployment data along with input data parameters for calculating discrete estimates for software maintenance, enhancement and growth effort. Input parameters are discussed as part of each strategy. Although source instructions were used as the size metric for the various strategies, functions points may also be used.

PRICE S Solution

A summary of PRICE S input parameters used for this case study are contained in Table G-1 of this appendix. The PRICE S economic assessment (spreadsheets) for an example strategy (retargeting) is summarized in Tables G-2, G-3, and G-4. Formulae, factors, elements of costs not estimated and unique assumptions are shown in the "Remarks" column. The process for accomplishing an economic assessment for reengineering with PRICE S starts with establishing an Estimating Breakdown Structure (EBS). The initial EBS should be for "Maintaining the Status Quo" software system or baseline system. Once input parameters for this system are established, completing the economic assessment becomes a task of changing only the inputs that are different from the baseline. For example, typically only three or four inputs were changed for each of the economic assessments accomplished for the sample case. Risk simulation using Monte Carlo or Latin Hybercube simulation techniques can then be accomplished on each of the input changes using one of or a combination of four distribution types: Normal, Triangular, Beta, and Uniform. This provides the capability to evaluate results in a statistical manner and perform risk abatement. Investment costs from each of the PRICE

S solutions may be directly mapped into the various worksheets depicted in Appendix B, Volume I of the SRAH.

Although multiple Estimating Breakdown Structures (EBS) could be developed, for this sample case the entire economic assessment was accomplished within one EBS. The estimating breakdown structure is displayed in Figure G-1. Establishing the PRICE S inputs for each element within the EBS, and which specific parameters require adjustments for accomplishing the economic assessment for the reengineering sample case will be covered in the following discussions.

PRICE S calculates up to nine software schedule milestones based on input parameters. Each calculated phase is reported on the output report. For the purpose of this report model calculated dates were used for predicting when software operations and support would start for each strategy (except for Maintain Status Quo).

G.3 Maintain Status Quo

The purpose of the "Maintain Status Quo" estimate is to establish the costs for the original software system. Figures G-2, G-3, G-4, and G-5 display the PRICE S inputs utilized for this estimate, which were obtained from the sample case overview. A short explanation is provided describing the source for each of the input parameters.

Figure G-2 depicts the development input parameters. The PLTFM input value selected is for a military ground software specification. This parameter reflects the specification level for the software, documentation, structure, reliability, and maintainability of the developed software. Coded values are provided for user's selection. The coded value of 1.2 reflects Military Ground software. Nominal values were selected for management complexity (*CPLXM*), internal integration (*INTEGI*), and hardware processor loading (*UTIL*) based on default table values.

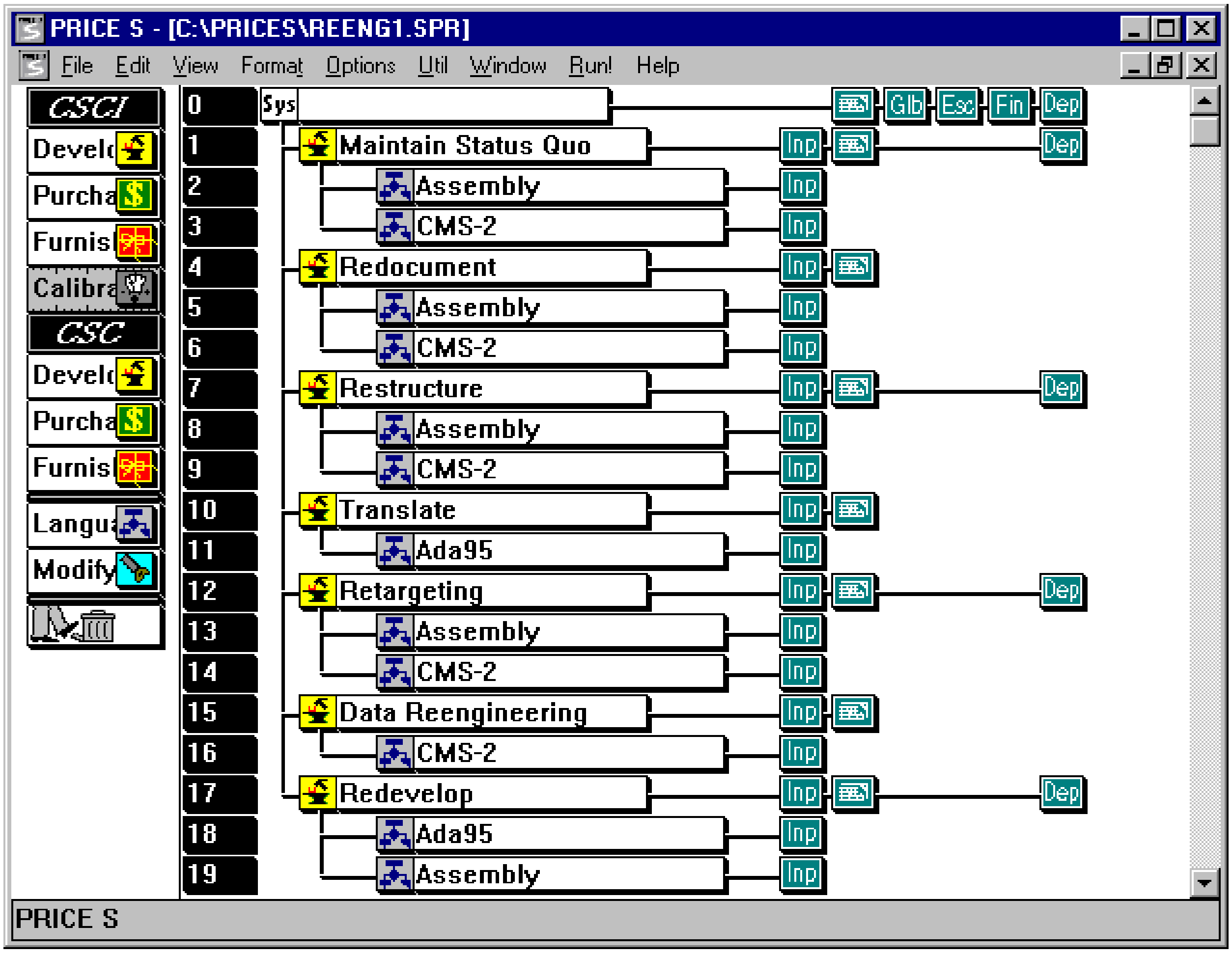


Figure G-1. PRICE S Estimating Breakdown Structure

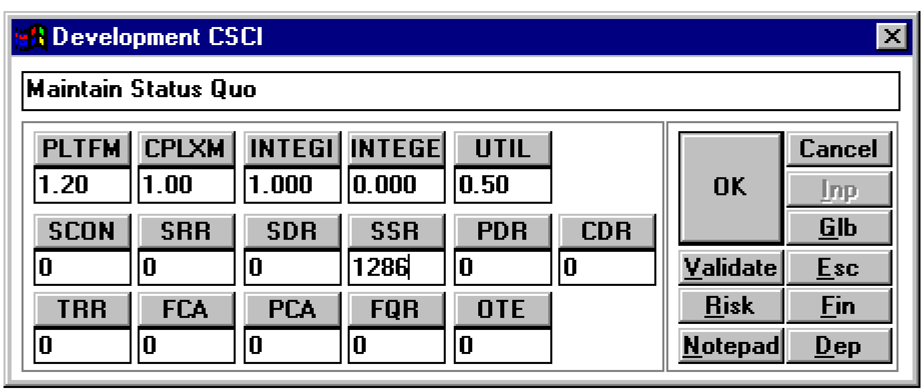


Figure G-2. PRICE S Development CSCI Inputs

External integration (*INTEGE*) was set to zero, since the software will not be integrated with other software packages within the EBS. A schedule start date of December 1986 was assumed for the SSR milestone. Schedule dates are entered into the PRICE S Model in MMYY format. Except for the SSR date, these parameters were held constant for each of the economic assessments.

PRICE S allows users to describe up to four languages per CSCI or component. This capability was used since two languages were identified in the sample case. Figure G-3 depicts the Assembly language input parameters. A nominal organization parameter (*PROFAC*) was selected from PRICE S guidelines. This is the calibratable parameter within the model that reflects the organization efficiencies, and inefficiencies and can be related to the Software Engineering Institute (SEI) Capability Maturity Model (CMM) levels. The number of source instructions are reflected in the input parameter SLOC. The input value of 3,174 for Assembly was obtained from the sample case description. The fraction of non-executable (*FRAC*) code was assumed to be ten percent. This input parameter reflects the percentage of "Type Statements," "Data Declarations," and "Format Statements." Development (*CPLX1*) and hardware/software integration (*CPLX2*) complexity factors were assumed to be nominal. The functional mix (*APPL*) for the assembly language was assumed to be for real time mux bus software. New design and code were assumed to be one hundred per cent.

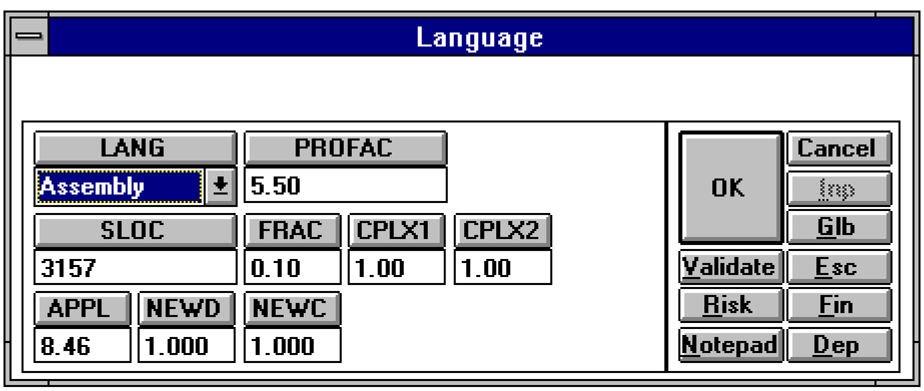


Figure G-3. PRICE S Language Inputs - Assembly

Figure G-4 depicts the CMS-2 language input parameters. A nominal organization parameter (*PROFAC*) was selected from PRICE S guidelines. This is the calibratable parameter within the model. The SLOC value of 28,417 for CMS-2 was obtained from the sample case study. The fraction of non-executable (*FRAC*) code was assumed to be ten percent. Development (*CPLX1*) and hardware/software integration (*CPLX2*) complexity factors were assumed to be nominal. The functional mix (*APPL*) for the CMS-2 software was assumed to be for application type software consisting of data base functions, format and buffer control functions, and mathematical functions. New design and code were assumed to be one hundred per cent.

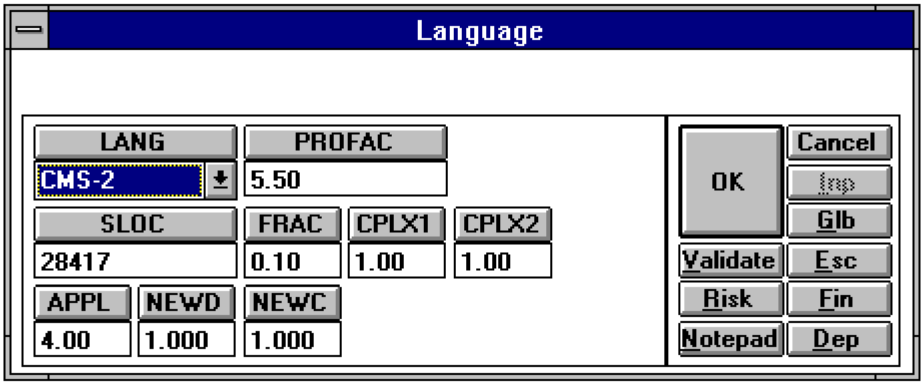


Figure G-4. PRICE S Language Inputs - CMS-2

The PRICE S Model calculates software support costs using the input data parameters described for each language and the deployment data depicted in Figure G-5. The deployment data contains support life parameters, growth, enhancement and quality level inputs, and productivity factors for each of the support phases.

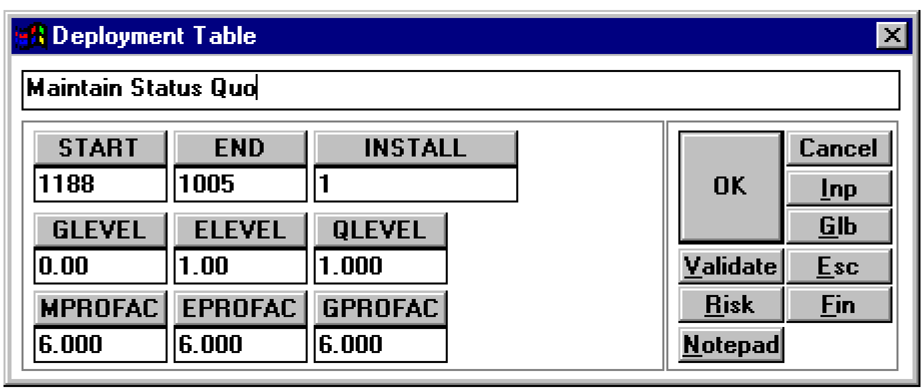


Figure G-5. PRICE S Deployment Inputs

A seventeen year support period was identified to the model by the dates indicated in the Start and End input parameters shown in Figure G-5. The number of software installations was assumed to be one and nominal values were used for enhancement and quality software levels, along with maintenance, enhancement and growth organization capabilities

(*MPROFAC*, *EPROFAC*, and *GPROFAC*).

Default or model provided financial rates were used for converting model calculated effort to currency. Currency output can be obtained in either constant year or as spent. The above input values were used to obtain the status quo estimate.

G-4 Redocument

The same inputs as the status quo were used for the redocument assessment, except new design and new code percentages were reduced to five percent, the repair level parameter was reduced to account for less time required to correct an error, and the start date was set to December 1996. Also a custom cost element table was established (*GBL*) allowing only software documentation costs to be calculated by the model. This table allows users to tailor the model to their estimating task.

G.5 Restructure

Figures G-6, G-7, G-8, and G-9 display the PRICE S inputs used for the restructure economic assessment. A short explanation is provided on parameters changed.

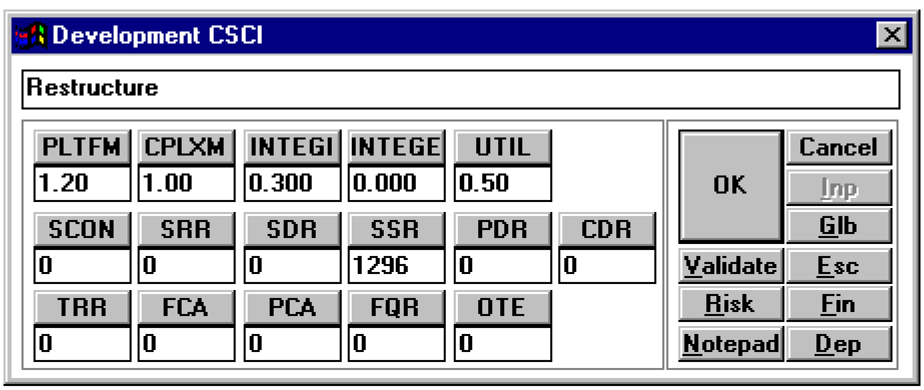


Figure G-6. Development CSCI Inputs - Restructure

The only inputs changed for the re-structure economic assessment were the start date and the value of the internal integration factor (*INTEGI*). A date of December 1996 was assumed to be the effort start date. The rationale for the integration adjustment is that integration should not be as difficult since this task was accomplished in the original effort.

The only input parameters requiring change were the development complexity (*CPLX1*), new design (*NEWD*) and code (*NEWC*). The development complexity was reduced to reflect the familiarity with the software product, and development tools.

New design and code values were reduced to 0.1 and 0.2 respectively, to reflect the design changes caused by restructuring. All other inputs were held constant. The quality level (QLEVEL) input parameter was increased to account for fewer errors being present in the re-structured software versus the originally developed software.

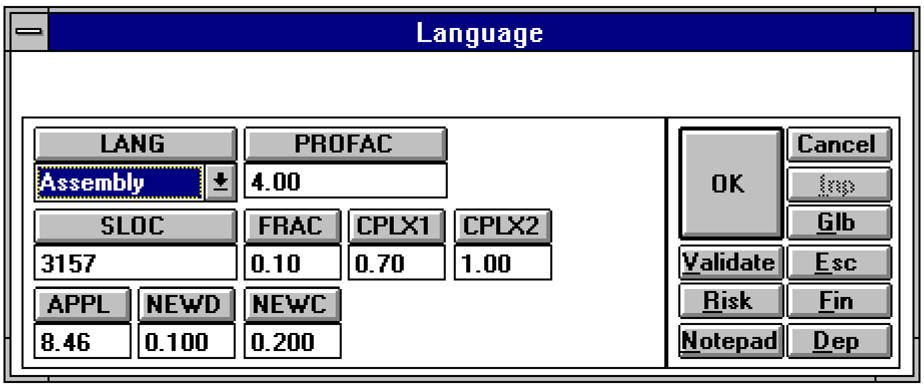


Figure G-7. Language Input Parameters for Assembly - Restructure

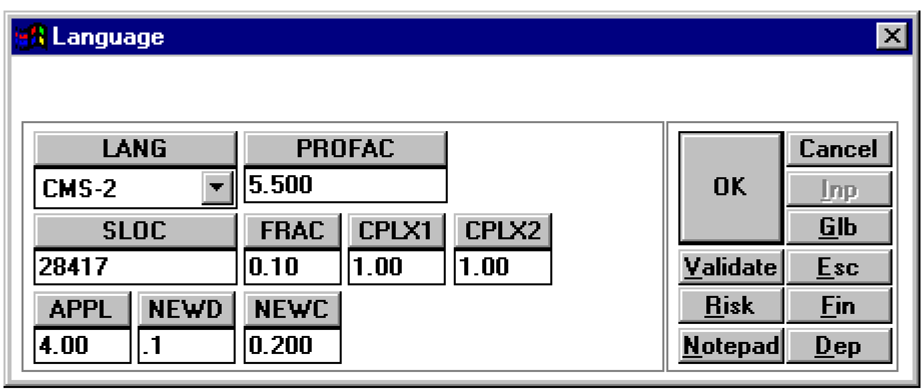


Figure G-8. Language Input Parameters for CMS-2 - Restructure

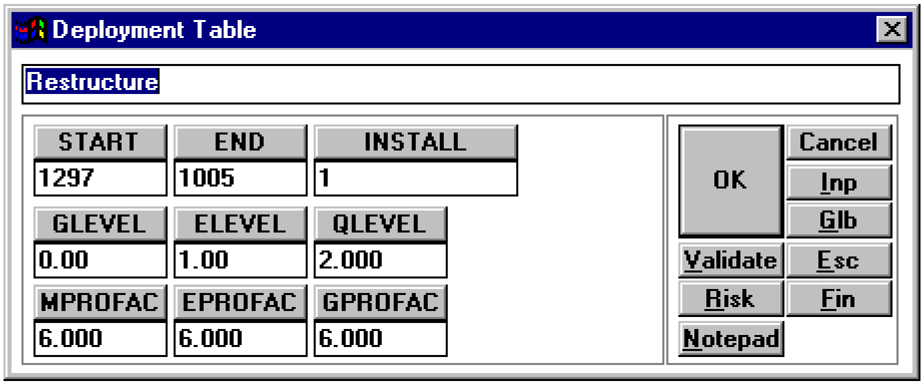


Figure G-9. Deployment Input Parameters - Restructure

G.6 Translation

Figures G-10 and G-11 display the PRICE S inputs used for the translation economic assessment. A short explanation is provided on input parameters changed.

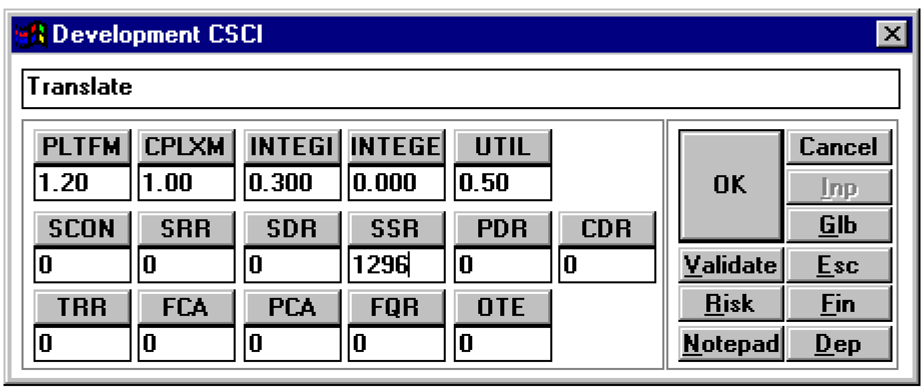


Figure G-10. Development CSCI Input Parameters - Translation

The only input parameter changed was the start date for the translation effort and the value of the internal integration factor. A date of December 1996 was assumed to be the effort start date. The internal integration value was reduced to account for less effort required for integrating code generated software.

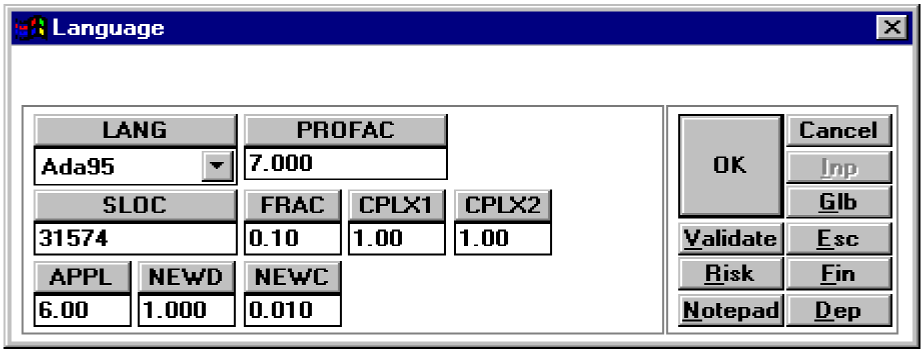


Figure G-11. Language Input Parameters - Translation

The case study identified *Ada* as the potential higher order language for the translation effort. No adjustments were made to *SLOC*, *FRAC*, *CPLX1*, *CPLX2* and *NEWD*. (It should be noted that SLOC should have been increased due to the translation, but for this case study it was held constant.) A higher value for *PROFAC* was used to account for the increased development efficiency for using Ada. The functional mix (*APPL*) was increased to account for additional software difficulties due to the combining the real time software (hardware processor dependent) with the previous application software. The additional weighting accounts for the fact that real time software is more expensive to develop and translate than mathematical or data array software functions. *NEWD* was set to 1.0 (100%), which assumes the translation effort will redesign the software in total to take advantage of the Ada language (this value could be set to a lower value if required). *NEWC* (new code) was set to 0.01 for modeling automatic code generation (this value reflects automatic code generation with no manual coding effort required).

G.7 Retargeting

Figure G-12, G-13, G-14 and G-15 depict the PRICE S inputs for the retargeting assessment. A short explanation is provided on each input parameter changed to model this strategy.

The only CSCI Project input parameters changed for this scenario was the start date for the retargeting effort and the value of the internal integration factor (*INTEGI*). A date of December 1996 was assumed for the start date. The internal integration value was reduced to account for less effort required for integrating the retargeted software. The rationale for reducing this input value is previous integration procedures and test plans can be reused.

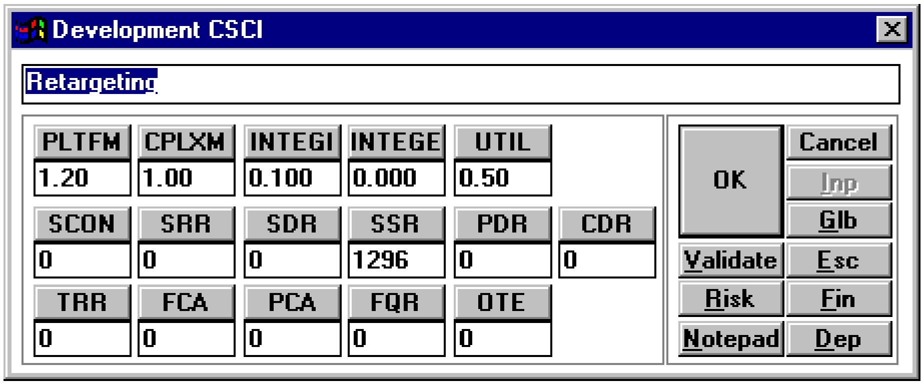


Figure G-12. Development CSCI Input Parameters - Retargeting

Development complexity (*CPLX1*), amounts of new design (*NEWD*) and new code (*NEWC*) were the only language input parameters changed. These changes were made to both languages (CMS-2 and Assembly).

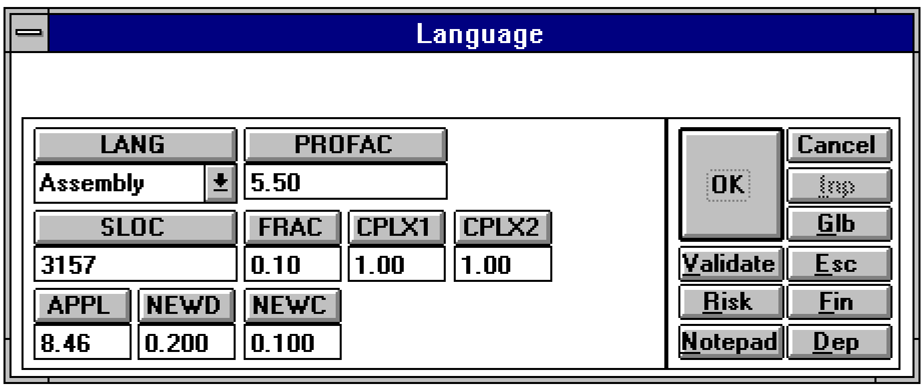


Figure G-13. Language Input Parameters - Retargeting

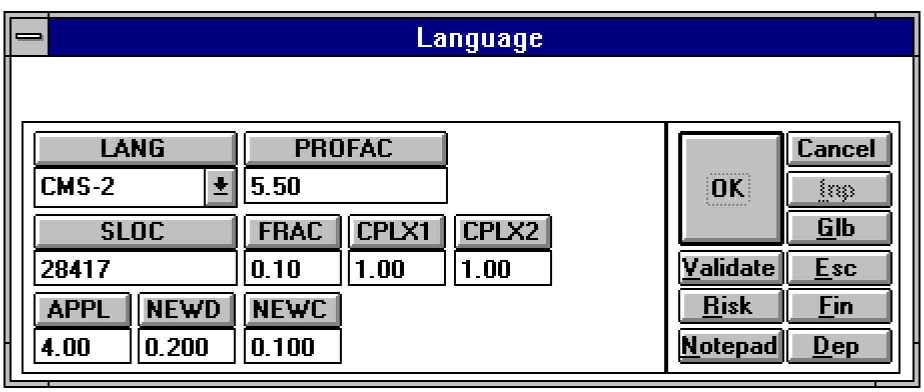


Figure G-14. Language Input Parameters - Retargeting

The rationale for changing the development complexity was the development team accomplishing this task should be familiar with the software design and code. New design was reduced to twenty percent to account for minor amounts of re-design required for rehosting the software. This value was selected to account for the effort required to accomplish a complete review of the design, and identifying areas of the software design that must be modified. New code was reduced to ten percent to account for effort to modify the code, due to the new configuration system.

It should be noted that the assumption for this strategy was the original software was developed to allow for rehosting. PRICE S can be used to model this type of reusability.

The only input changed for support was quality level (QLEVEL). This parameter was increased slightly to account for the number of defects that were corrected during the retarget process. Increasing this parameter results in less defects calculated and reported by the model.

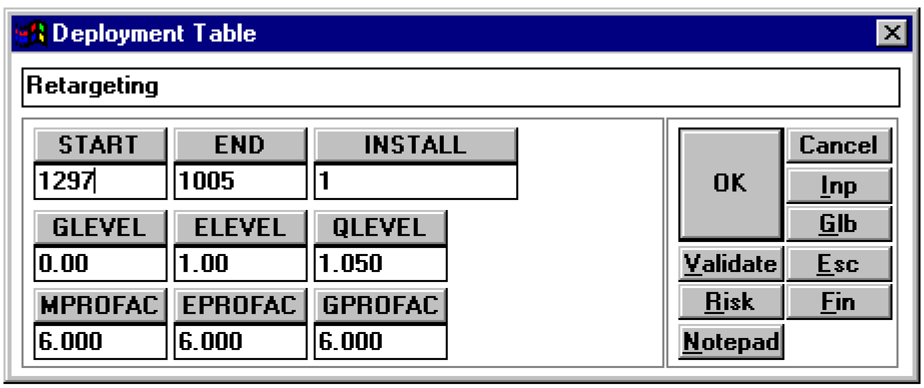


Figure G-15. Deployment Input Parameters - Retargeting

G.8 Data Reengineering

Figure G-16, G-17, and G-18 depict the PRICE S inputs for the data reengineering assessment. A short explanation is provided on each input parameter changed to model this strategy.

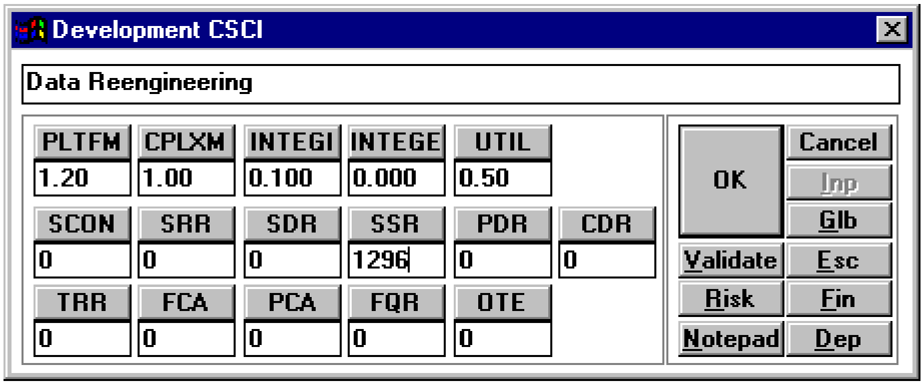


Figure G-16. Development CSCI Input Parameters - Data Reengineering

The only CSCI Project input parameters changed for this scenario was the start date for the retargeting effort and the value of the internal integration (*INTEGI*) factor. A date of December 1996 was assumed to be the effort start date. The internal integration value was reduced to account for less effort required for integrating the retargeted software. The rationale for reducing this input value is previous integration procedures and test plans can be reused.

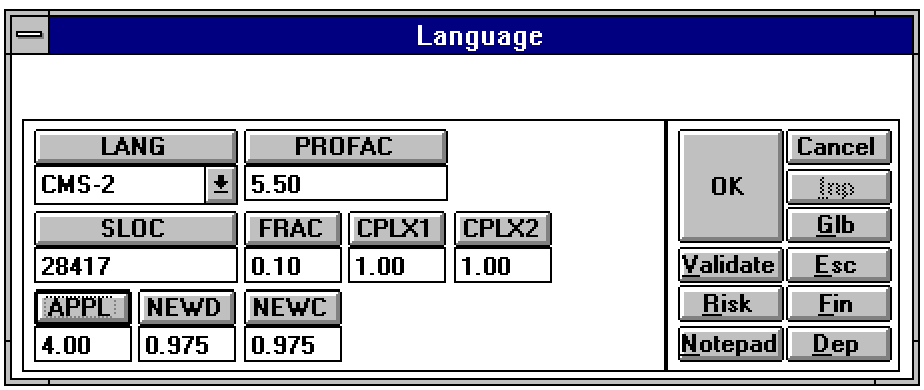


Figure G-17. Language Input Parameters - Data Reengineering

The language inputs changed consisted of the Data Storage and Retrieval functional mix input parameter. This parameter is located in the "*Application Generator*" window shown in Figure G-18. PRICE S provides the capability to model the functional mix of the input source instructions. The empirical weights are assigned to each functional mix and are used to calculate a weighted average. This weighted application average is also used for new design and new code to account for the fact that some portions of a software system are more difficult to design and implement than others. For example, when estimating costs for a system in which half of the code is to be reused, simplistic reasoning might lead to a conclusion that the effort required would be about half that for one-hundred percent new code. This reasoning would fail to recognize that it is almost invariably inexpensive code (utilities math, etc.) that is available for reuse, and that the difficult, machine dependent and problem specific code has yet to be constructed. Application-weighted averages avoid this pitfall.

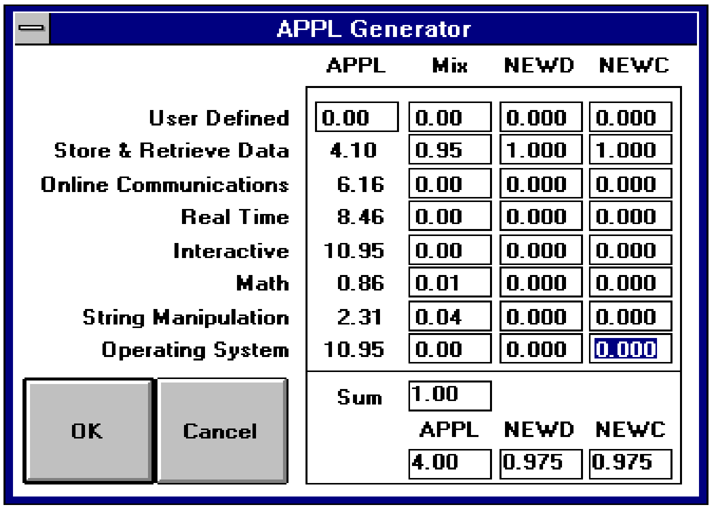


Figure G-18. Application Generator - Data Reengineering

Support input parameters for data reengineering were the same as the retargeting assessment.

G.9 Reverse Engineering

Reverse engineering can be modeled in PRICE S using two steps. The first step is to model the effort or process of obtaining an understanding, analysis, and abstracting the existing system to a new form at a higher abstraction level. This is accomplished by describing input parameters to model for the software system that is being considered for reverse engineering. A custom program constant element table is also required. This table allows for tailoring the model estimate to the reverse engineering tasks. In this case all software phases were eliminated from the estimate except for software requirement analysis. The effort depicted from this phase is only for the reverse engineering process. Figure G-19 depicts the tailored program constant element table, and figure G-20 depicts the resulting model output. Inputs used were the same as the “Maintain Status Quo” scenario.

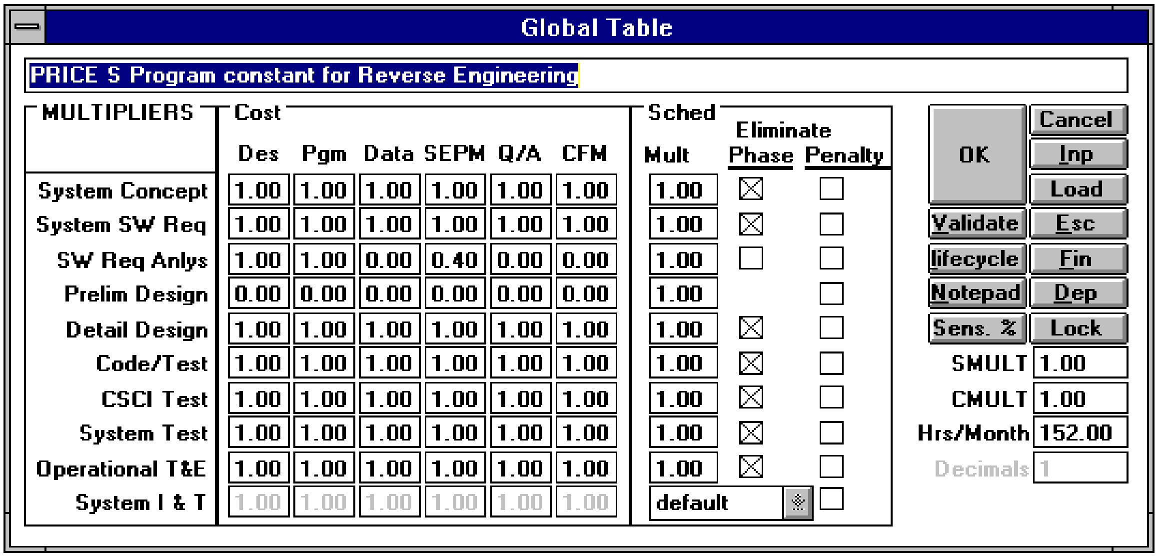


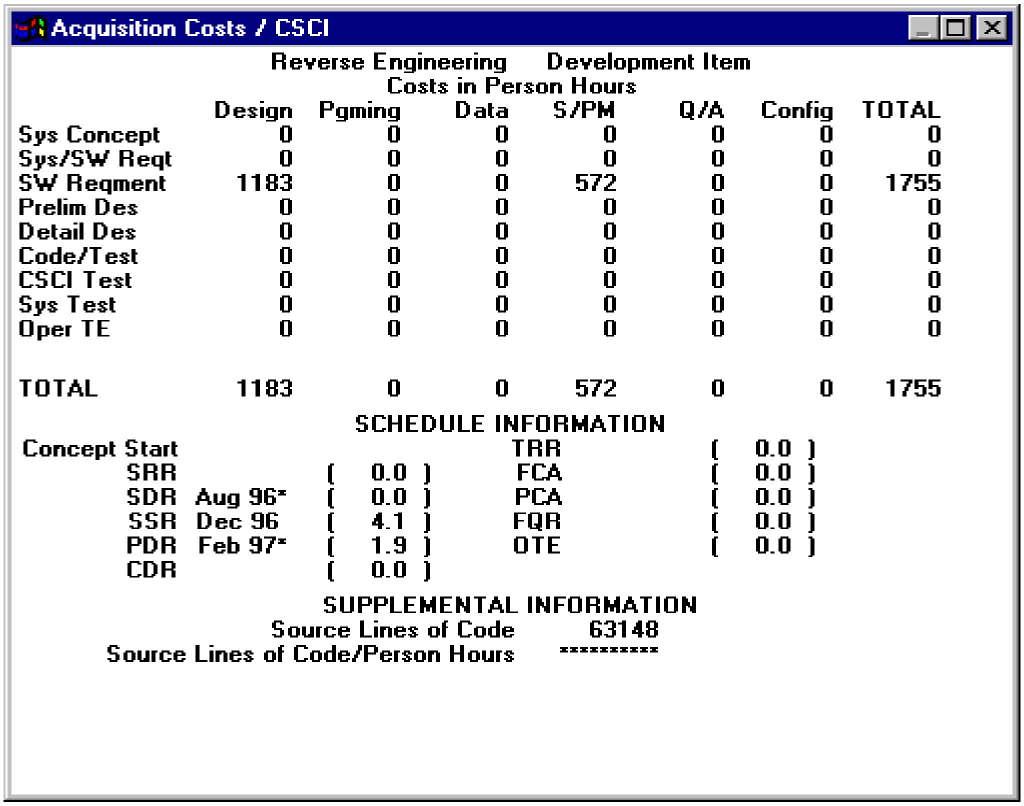
Figure G-19. PRICE S Program Constant Table for Reverse Engineering

The second step consists of describing the results of step one to the model. A different program constant table would be used for this step, which only accounts for design, code and test. This is accomplished by simply removing the check from applicable phase elimination boxes. The only other changes required would be in the operations and support element, where the quality level input (*QLEVEL*) should be increased slightly to account for less defects in the software.

The reason for a two step model process is this allows the user to differentiate between different reverse engineering approaches, rather than have a model simply complete an estimate.

G.10 Combination of Reengineering Strategies

Applying this process would consist of step one from the previous discussion on reverse engineering, and then using any of the candidate strategies as step two.



Acquisition Costs / CSCI

Reverse Engineering Development Item

Costs in Person Hours

	Design	Pgming	Data	S/PM	Q/A	Config	TOTAL
Sys Concept	0	0	0	0	0	0	0
Sys/SW Reqt	0	0	0	0	0	0	0
SW Reqment	1183	0	0	572	0	0	1755
Prelim Des	0	0	0	0	0	0	0
Detail Des	0	0	0	0	0	0	0
Code/Test	0	0	0	0	0	0	0
CSCI Test	0	0	0	0	0	0	0
Sys Test	0	0	0	0	0	0	0
Oper TE	0	0	0	0	0	0	0
TOTAL	1183	0	0	572	0	0	1755

SCHEDULE INFORMATION

Concept Start			TRR	[0.0]
SRR		[0.0]	FCA	[0.0]
SDR	Aug 96*	[0.0]	PCA	[0.0]
SSR	Dec 96	[4.1]	FQR	[0.0]
PDR	Feb 97*	[1.9]	OTE	[0.0]
CDR		[0.0]		

SUPPLEMENTAL INFORMATION

Source Lines of Code 63148

Source Lines of Code/Person Hours **********

Figure G-20. PRICE S Output Report for Reverse Engineering

G.11 Redevelopment

Figures G-21, G-22, G-23 and G-24 display the PRICE S inputs used for the re-developed economic assessment. A short explanation is provided on parameters changed.

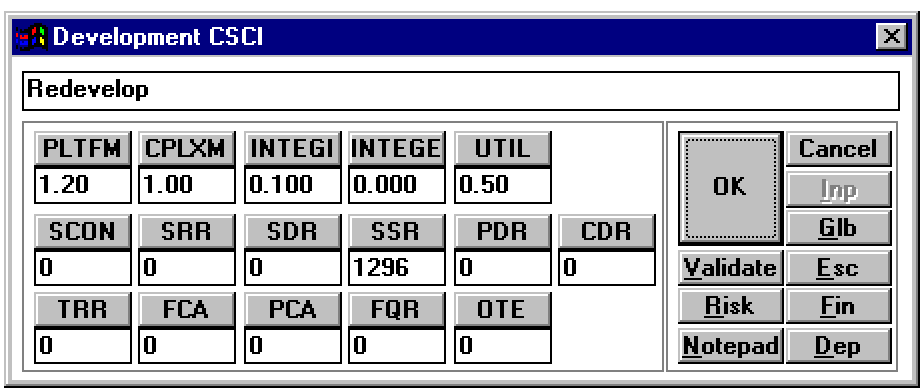


Figure G-21. Development CSCI Input Parameters - Redevelop

The same development CSCI input parameters used for the restructure assessment were used for the redevelop assessment.

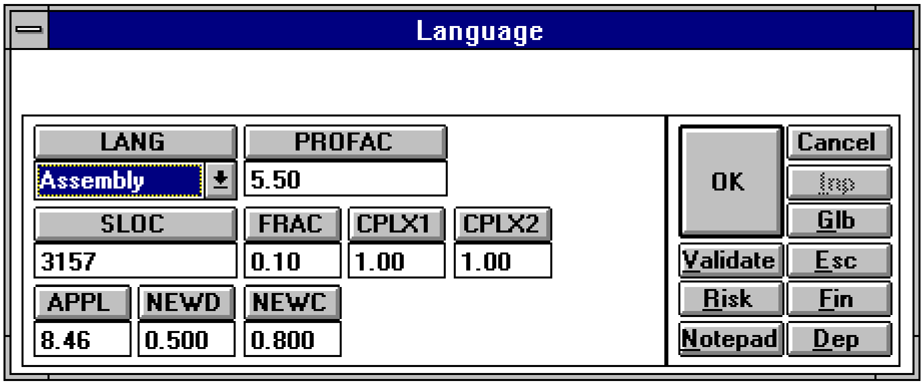


Figure G-22. Language Input Parameters for Assembly - Redevelop

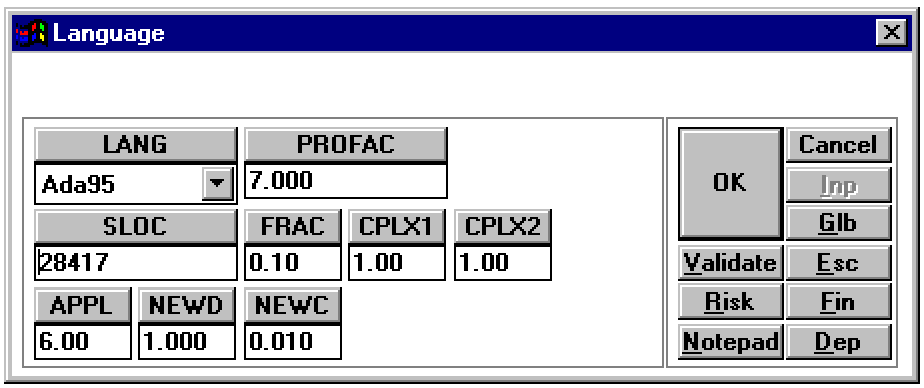


Figure G-23. Language Input Parameters for Ada - Redevelop

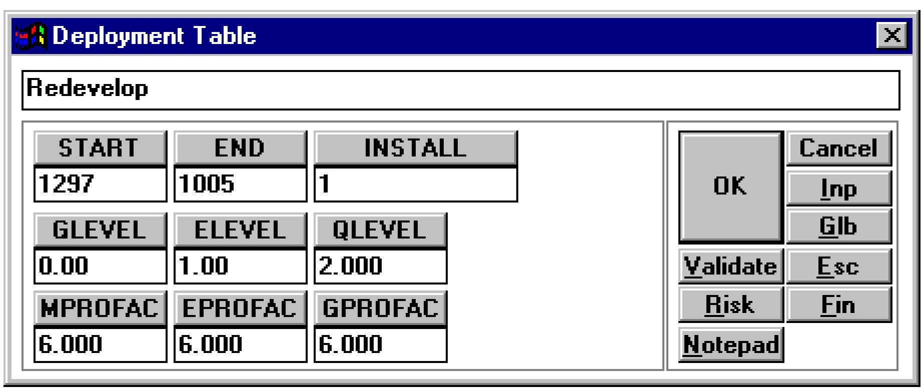


Figure G-24. Deployment Input Parameters - Redevelop

The only input parameters changed for the language assessment were new design and code. These input values were obtained from the sample case discussion on redevelopment.

The quality level (QLEVEL) input parameter was increased to account for fewer errors being present in the redeveloped software versus the originally developed software.

Risk simulation was accomplished on the restructuring economic assessment. A triangular distribution was established for SLOC. The minimum and maximum values entered were calculated using an ancillary program provided with the model. These values were then processed through the risk simulation model using one thousand simulations. The results are depicted in Figure G-25 for acquisition and Figure G-26 for life cycle cost.

Tables G-1, G-2, G-3, and G-4 depict PRICE S results for the retargeting strategy. This strategy will be used as an example for completing the Reengineering Assessment Economic Worksheets. Note: The same support life was used for all strategies for this evaluation. Different support life periods could have been entered and used as part of this evaluation. Also in general the support life costs for each reengineering strategy reflected lower costs than the status quo.

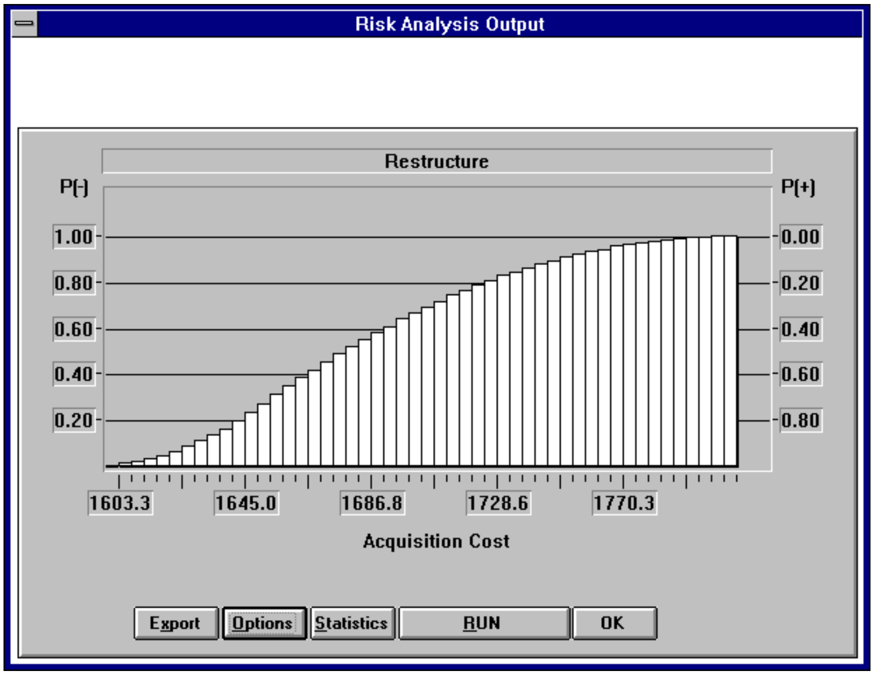


Figure G-25. PRICE S Cumulative Distribution Function for Acquisition - Restructure

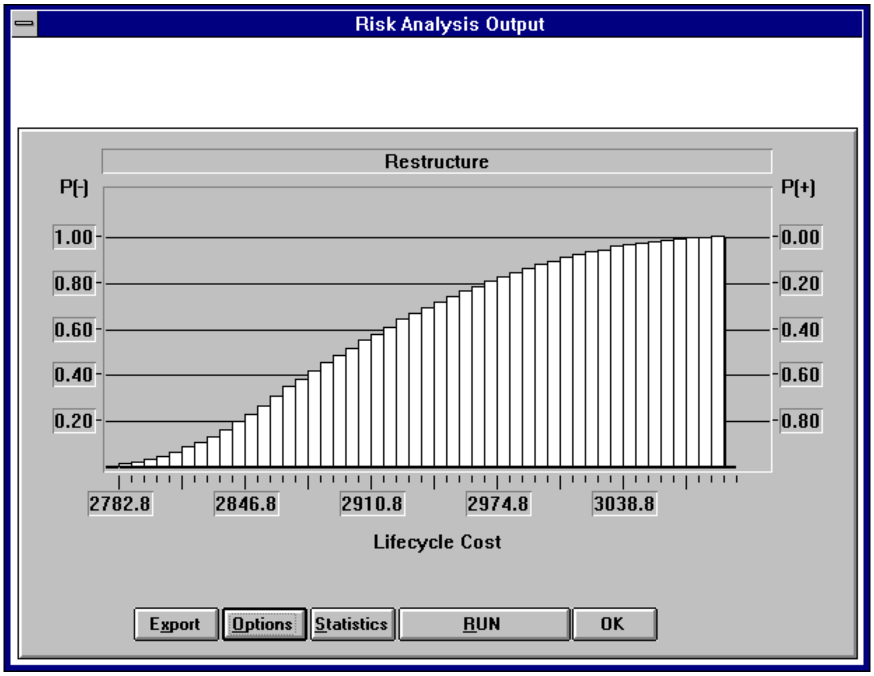


Figure G-26. PRICE S Cumulative Distribution Function for Life Cycle Cost - Restructure

Parameter/CES/Indicator	Status Quo	redocument	Translate	Restructure	Redevelop	Remarks
A. Parameters						
PLTFM	1.2	1.2	1.2	1.2	1.2	Specification Level
Assembly SLOC	3157	3157	0	3157	3157	SLOC used for Assy
APPL	8.46	8.46	0	8.46	8.46	Functional mix for Assy
CMS-2 SLOC	28417	28417	0	28417	0	SLOC used for CMS-2
APPL	4.00	4.00	0	4.00	0	Functional mix for CMS-2
ADA SLOC	0	0	31574	0	28417	SLOC used for ADA
APPL	0	0	6.00	0	4.00	Functional mix for ADA
New Design	1.00	0.05	1.00	0.10	0.50	Sample case Description
New Code	1.00	0.05	0.01	0.20	0.80	Sample case Description
PROFAC	5.50	5.50	5.50	5.50	5.50	Empirical value.
CPLX1	1.00	1.00	1.00	0.70	1.00	Dev. personnel and tools
CPLX2	1.00	1.00	1.00	1.00	1.00	HSI complexity
INTEGI	1.00	1.00	0.30	0.30	0.30	Integration difficulty
UTIL	0.50	0.50	0.50	0.50	0.50	Timing & memory parameter
QLEVEL	1.00	1.00	1.00	2.00	2.00	Quality of Dev. S/W.

Table G-1. PRICE S Input Parameter Summary

CES Cost Element	FY96	FY97	FY98	FY99	FY00	FY01	FY02	FY03	FY04	FY05	Total	Data Source
1.1 Software Dev.												Model Calculated
1.1.1 Reqt. Anal	24.0										24	Model Calculated
1.1.2 Prel Des.	46.2	0									46.2	Model Calculated
1.1.3 Detatiled Des	27.8	41.6									69.4	Model Calculated
1.1.4 Code/Unit Tst		19.7									19.7	Model Calculated
1.1.5 Unit Integ&Tst		29.5									29.5	Model Calculated
1.1.6 CSCI Test		94.8									94.8	Model Calculated
1.1.7 SPCR Res.												
1.1.8 Other												
1.2 CSCI-CSCI I&T												Modeled as 1 CSCI
1.3 Sys I&T												Modeled as 1 CSCI
1.4 Training												
1.5 Data	4.6	21.1									25.7	Model Calculated
1.6 Pec. Spt Eq.												Cal. By PRICE H
1.7 Ops Site Act.												
1.8 Fac. & Ut.												
1.9 Hardware												Cal By PRICE H
1.10 System Oper.												
1.11 IV&V												
1.12 Sys Eng/PM	45	45									90	Model Cal
1.13 Other												HW LCC calculated by PRICE HL
1.0 Investment (Constant $)	147.6	251.7									399.3	
Mid-Yr Dis. Factor												
1.0 Investment (Present Value)												
Interest Rate												
1.0 Investment (Current $)												

Table G-2. Reengineering Investment Cost - Retargeting Strategy

CES Cost Element	FY96	FY97	FY98	FY99	FY00	FY01	FY02	FY03	FY04	FY05	Total	Data Source
2.1 Software Spt.			68.4	85.6	78.0	60.4	41.0	25.0	15.1	9.1	382.6	Model Cal. Retar.
2.1.1 Reqt. Anal												
2.1.2 Prel Des.												
2.1.3 Detatiled Des												
2.1.4 Code/Unit Tst												
2.1.5 Unit Integ&Tst												
2.1.6 CSCI Test												
2.1.7 SPCR Res.												
2.1.8 Other												
2.2 CSCI-CSCI I&T												
2.3 Sys I&T												
2.4 Training												
2.5 Data												
2.6 Pec. Spt Eq.												Cal. By PRICE H
2.7 Ops Site Act.												
2.8 Fac. & Ut.												
2.9 Hardware												Cal By PRICE H
2.10 System Oper.												
2.11 IV&V												
2.12 Sys Eng/PM												
2.13 Other												HW LCC calculated by PRICE HL
2.14 Status Quo O&S	250	250	0	0	0	0	0	0	0	0	500	Model Calculated
2.0 O&S (Constant $)	250	250	68.4	85.6	78.0	60.4	41.0	25.0	15.1	9.1	882.6	Model Calculated
Mid-Yr Dis. Factor												
2.0 O&S (Present Value)												
Interest Rate												
2.0 O&S(Current $)												

Table G-3. Reengineering Operations and Support Cost - Retargeting Strategy

CES Cost Element	FY96	FY97	FY98	FY99	FY00	FY01	FY02	FY03	FY04	FY05	Total	Data Source
3.1 Software Spt.												Model Cal. Retar.
3.1.1 Reqt. Anal												
3.1.2 Prel Des.												
3.1.3 Detatiled Des												
3.1.4 Code/Unit Tst												
3.1.5 Unit Integ&Tst												
3.1.6 CSCI Test												
3.1.7 SPCR Res.												
3.1.8 Other												
3.2 CSCI-CSCI I&T												
3.3 Sys I&T												
3.4 Training												
3.5 Data												
3.6 Pec. Spt Eq.												Cal. By PRICE H
3.7 Ops Site Act.												
3.8 Fac. & Ut.												
3.9 Hardware												Cal By PRICE H
3.10 System Oper.												
3.11 IV&V												
3.12 Sys Eng/PM												
3.13 Other												HW LCC calculated by PRICE HL
3.0 O&S (Constant $)	250	250	250	250	250	250	250	250	250	250	2500	Model Calculated
Mid-Yr Dis. Factor												
3.0 O&S (Present Value)												
Interest Rate												
3.0 O&S (Current $)												

Table G-4. Status Quo Operations and Support Cost

APPENDIX H

SEER-SEM SOFTWARE ESTIMATING MODEL

This appendix describes how the SEER-SEM software estimating model can be applied to reengineering efforts. It was prepared by Ms. Karen McRitchie and Mr. Alan Clark of Galorath Associates and edited by Mr. John Clark of Comptek Federal Systems.

Ms. Karen McRitchie
Mr. Alan Clark
Galorath Associates, Inc.
SEER Technologies Division
100 N. Sepulveda Boulevard, Suite 1801
El Segundo, CA 90245
(310) 414-3222 (Voice)
(310) 414-3220 (Fax)
info@gaseer.com
www.gaseer.com

H.1 Introduction

Galorath Associates' System Evaluation and Estimation of Resources - Software Estimation Model (SEER-SEM) was designed for ease of use and flexibility. Its features make it suitable to estimate the cost and schedule of software reengineering projects like restructuring, retargeting, and translation, as well as maintenance only, incremental build, and new development. SEER-SEM is a parametric cost, effort, maintenance, risk, and schedule estimating tool. Its primary inputs are software size (in lines of code or function points), personnel factors, and environmental factors. Estimators using SEER-SEM may do exhaustive research to load all parameter values manually, or may rely on the program itself for some inputs via the use of *Knowledge Bases*.

Like the other SEER models, SEER-SEM derives its user friendliness from its use of knowledge bases. These k-bases are repositories of historically valid values for software development categories such as platform, application, acquisition method, development method, and development standard. *Platform* describes the primary mission and/or operating environment of the software to be estimated such as avionics, business, manned space, missile, mobile, ship, or unmanned space. *Application* describes the primary function of the software such as CAD, database, diagnostics, flight, graphics, MIS, simulation, testing, training, utilities, etc. *Acquisition Method* describes the scope and type of project being developed or maintained. As this category of knowledge base speaks directly to reengineering, a detailed glossary of acquisition method k-bases is included in section H.2, Model Input Parameters. *Development Method* is the paradigm or approach used for the project. Examples of SEER-SEM development method knowledge bases include Ada, commercial off-the-shelf, object oriented, prototype, spiral, evolutionary, traditional incremental, and waterfall. Four Ada development method knowledge bases exist. ADAFULL is exactly what the name implies, covering full use of the Ada programming language with development tools and methods. ADAINC represents a development team not dedicated to Ada. It covers the use of Ada as a programming language following a modern incremental development process. ADADEV covers the use of Ada as a programming language only, with no dedicated tools, practices, or methods. ADA-OO combines the use of the Ada programming language with object oriented design and programming methods. *Development Standard* addresses specification, test, and quality assurance requirements. Examples include commercial, ANSI J-STD-016 (full, nominal, minimal), FAA, IEEE, ISO 9001, DoD 1703, DoD 2167A (full, minimal, tailored), DoD 7935, Mil-Std 498 (business, support, & weapon systems), and Navy 1679 with IV&V.

SEER-SEM knowledge bases allow users to produce an estimate with only size and basic program description inputs, because the k-bases can provide the rest of the information needed. Knowledge bases load a series of parameters in a range of *least* (or lowest possible), *likely*, and *most* (or highest possible) values from which an expected value is calculated in a pert probabilistic distribution. Users with known parameter inputs can refine their estimates by replacing k-base values with their own inputs. The least, likely, and most input ranges are especially useful for performing what-if or sensitivity analyses.

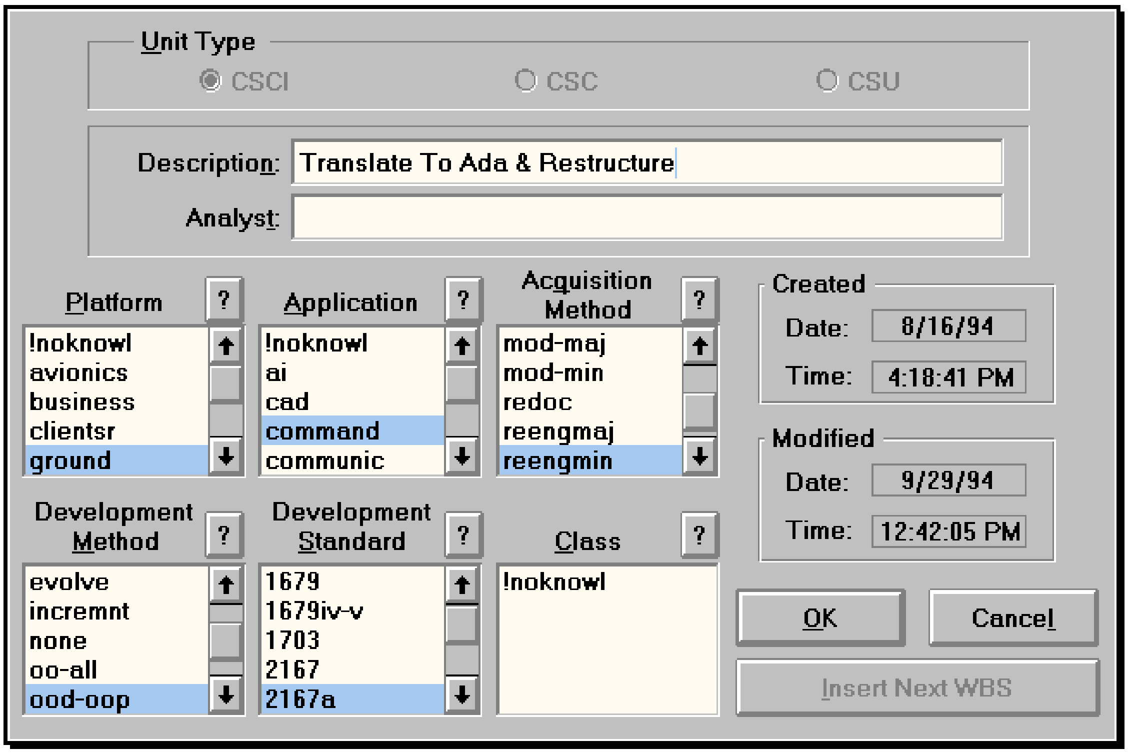


Figure H-1: Selecting Knowledge Bases With SEER-SEM

Estimators using SEER-SEM for a reengineering project (or any other, for that matter), would begin by entering the project work breakdown structure (WBS). SEER-SEM accepts line items at the project, rollup, CSCI, CSC, and CSU levels. To create a new project file, the user selects "File New" from the menu bar. After typing the project title and analyst names, click on the "Insert Next WBS" button and repeat the process for each CSCI by typing titles and selecting knowledge bases (see Figure H-1 above). After the last WBS item has been described, click on "OK." The user is then presented with four windows on the screen (see Figure H-2). *Project WBS* on the upper left is self explanatory and shows the CSCIs entered. *Parameter View* on the upper right shows the parameters for the WBS item currently highlighted in the Project WBS window. At this point, the user should click on the Parameter View window, where he can enter size in lines of code or function points (see Figure H-3). After entering new and/or pre-existing counts, the preliminary estimate is done. For most users, the least, likely, and most input ranges are a helpful way to bound risk of parameter inputs. Users who are certain of a parameter value can enter the value in all three fields, effectively creating a 100% confidence interval or zero risk for that input.

The lower left window is the report window. It defaults to the *Quick Estimate* report, showing key outputs including Development Schedule, Development Effort, Development Base Year Cost, and Defect Prediction. The Quick Estimate report is user customizable to show different outputs. The lower right window is the chart window. It defaults to *Estimate Assessment*, a bar chart displaying the reasonableness of parameter inputs in red, yellow, and green signifying questionable, marginal and reasonable inputs, respectively. The parameter categories displayed in the Estimate Assessment chart are size, technology, complexity, staffing, schedule, and probability.

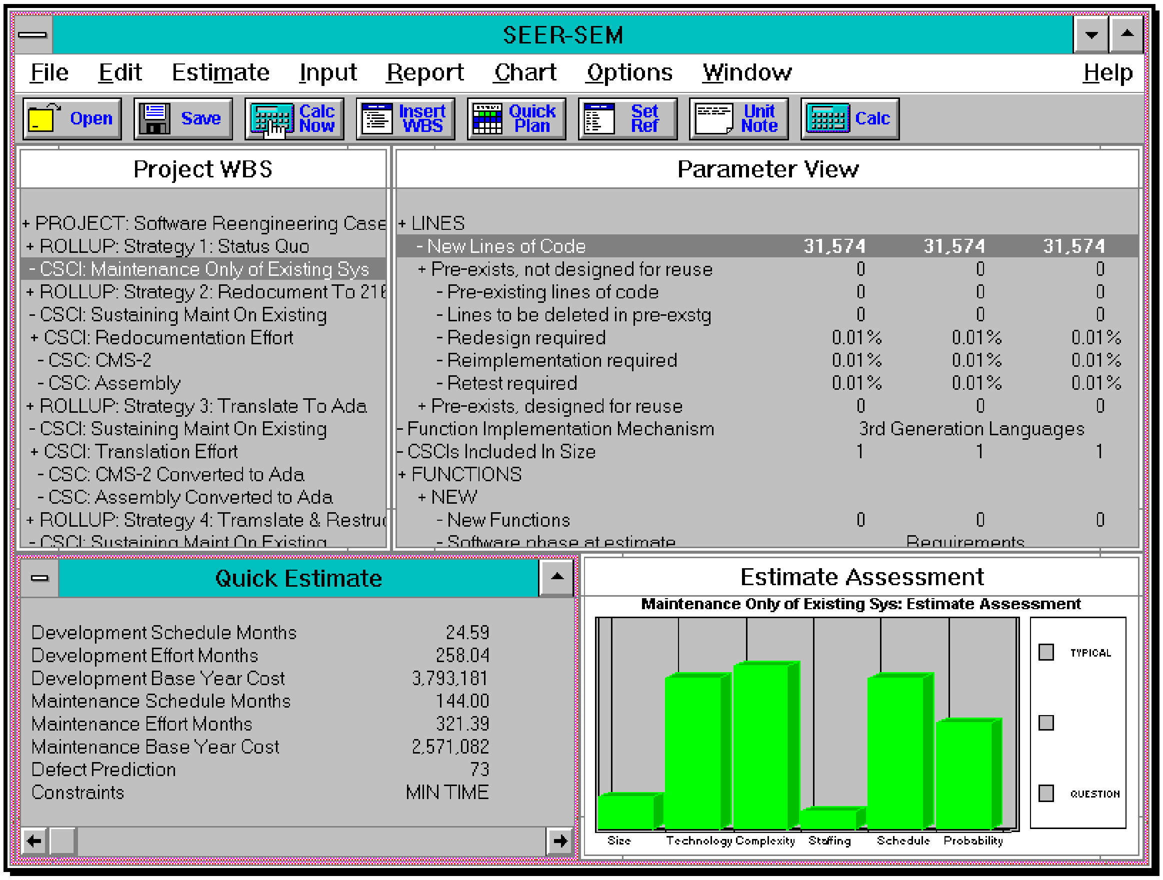


Figure H-2: Standard Four Window SEER-SEM Screen Layout

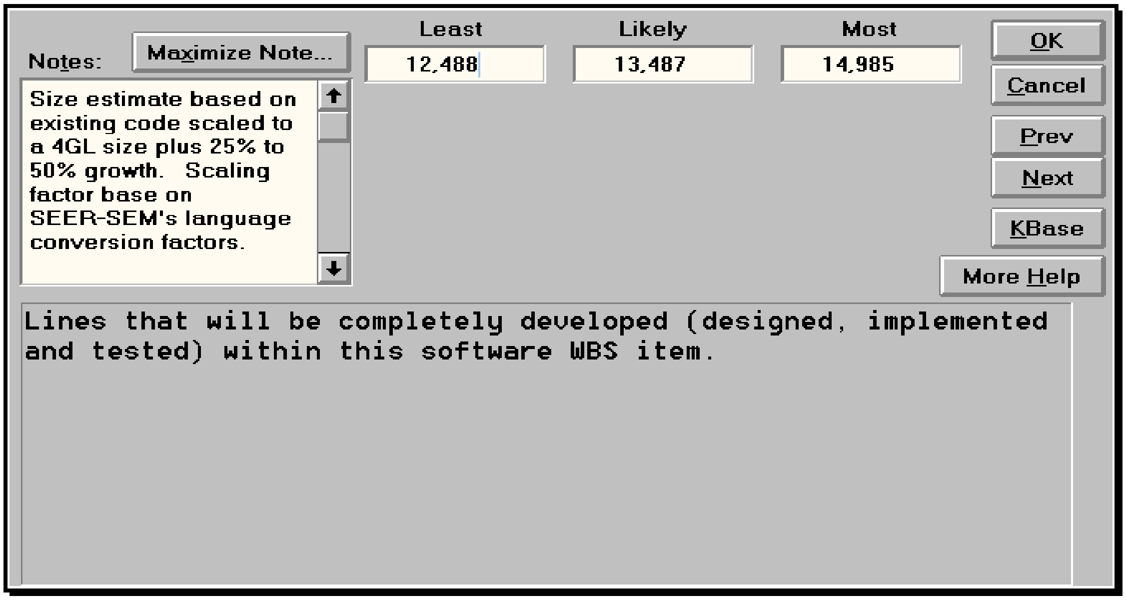


Figure H-3: Entering Parameter Ranges With SEER-SEM

H.2 Model Input Parameters

SEER-SEM has many knowledge bases which capture the characteristics of most types of software development projects, including the reengineering strategies identified in this case study. As mentioned previously, most of the reengineering specific knowledge bases are found in the *Acquisition Method* category. These knowledge bases provide guidance for sizing rework of preexisting functions or source lines of code. Specific Acquisition Method k-bases are:

New Development (~NEW). For development of a new system or when there is some preexisting software available. Should be used for the initial deliverable build of an incremental development. Choose ~NEW for the most general knowledge base in this category.

Subsequent Incremental Build (BUILD23). Use for a deliverable incremental build other than the initial build. This knowledge base assumes no requirements work after baseline. Maintenance estimates of subsequent builds are only for the portion of software being added and modified in this build.

Major Modification (MOD-MAJ). This describes a major modification to existing software. Typically, the existing software is being used for a new application or mission. Often it involves a target environment change. It assumes the programming language will not have any significant changes.

Minor Modification (MOD-MIN). This describes a minor modification to existing software. Typically, the existing software is being used for the same mission, but with some changes in functionality. The target environment and programming language will not have any significant changes.

Major Rehost (RHOSTMAJ). This describes rehosting software from one target environment to another. It assumes a change in operating systems, target hardware, and development tools. For example, a port of an application from Windows to a Macintosh environment. This knowledge base assumes the basic functionality of the application will remain intact, and the programming language will stay the same.

Minor Rehost (RHOSTMIN). This describes rehosting software from one target environment to a similar environment. It assumes no major operating system changes. Development tools may be different between the platforms. An example would be a port from an SGI Unix workstation to a Sun Unix workstation. This knowledge base assumes the basic functionality of the application will remain intact, and that the programming language will stay the same.

Major Reengineering (REENGMAJ). This describes major rework of an existing application to improve program structure, documentation, and maintainability. This effort will include moderate amounts of reverse engineering to ascertain the program design. This knowledge base assumes the basic functionality of the application will remain intact, and that the programming language will stay the same.

Minor Reengineering (REENGMIN). This describes minor rework of an existing application to improve program structure, documentation, and to a limited extent, maintainability. This knowledge base assumes the basic functionality of the application will remain intact, and that the programming language will stay the same.

Full Complete Maintenance (MAINTFUL). Use this knowledge base for maintenance estimates without development effort. Size should be entered in preexisting source lines or functions. This knowledge base assumes complete rigorous software maintenance including corrective, adaptive, & perfective maintenance and enhancements for 10 years. Users may change the maintenance period by changing the “Years Of Maintenance” parameter.

Sustaining Maintenance (MAINTMIN). Using this knowledge base will estimate sustaining maintenance for modifications and additions to the defined system. This maintenance effort will cover only the essential corrective and adaptive maintenance effort for 10 years. Perfective maintenance and enhancements are not included. Users may change the maintenance period by changing the "Years Of Maintenance" parameter.

Redocumentation (REDOC). This knowledge base is for estimating the effort required to make major revisions to the software specifications and manuals. No change is made to the software. Assumes some familiarity with the software and that some existing documentation (up to 25%) can be used. Note: this will impact the preexisting software only. New components will require normal development.

Manual Language Conversion (CONVMANL). Use this knowledge base for estimating the effort required to manually convert software from one language to another (e.g. FORTRAN to Ada). Assumes no change in the software design beyond what is dictated by the language change. The basic application and mission will remain intact. Note: this will impact the preexisting software only. New components will require normal development.

Automated Language Conversion (CONVAUTO). Use this knowledge base for estimating the effort required to convert software from one language to another (e.g. FORTRAN to Ada) using an automated tool. Assumes no change in the software design beyond what is dictated by the language change. The basic application and mission will remain intact. Note: this will impact the preexisting software only. New components will require normal development.

The SEER-SEM parameters most relevant to reengineering projects are redesign, reimplementation, and retest. They are set by whichever Acquisition Method knowledge base is selected by the user, and are, of course, user refinable. For users seeking guidance in adjusting the k-base supplied values for these parameters, the SEER-SEM User's Manual provides tables for calculating least, likely, and most inputs.

SEER-SEM's k-bases were used to determine the rework necessary for the various reengineering strategies and to set up default values for all technology and environment parameters. The selected k-base and other input parameter values can be found on the SEER-SEM *Inputs With Notes* report (see Figure H10). This report includes explanations for any specific parameter value that differed from the k-base supplied value as well as any general estimation ground rules and assumptions. Highlights of the key inputs are listed in the input table below (Figure H-4). A detailed explanation of the outputs follows the Econonmic Assessment Summary Table (Figure H-5).

Option Number	1	2	3	4	5
Strategy	Status Quo	Redocument	Translate Source Code	Translate & Restructure	Redevelop
Platform	Ground	Ground	Ground	Ground	Ground
Application	Command	Command	Command	Command	Command
Acquisition Method	Maintful	Redoc	Convauto	Reengmin	New
Development Method	Waterfal	Waterfal	Adadev	OOD-OOP	OO-ALL
Dev't Standard	483-490	2167A	2167A	2167A	2167A
Redesign	0%	29%	6%	31%	10%
Reimplementation	0%	0%	5%	20%	5%
Retest	0%	0%	49%	88%	40%
Language	CMS-2/ Assembly	CMS-2/ Assembly	Ada	Ada	4GL
Risk Level	50%	50%	50%	50%	50%
Development Labor Rate/Mo.	$14,700	$14,700	$14,700	$14,700	$14,700
Maintenance Labor Rate/Mo.	$8,000	$8,000	$8,000	$8,000	$8,000
Years of Maintenance	12	11	11	11	11

Figure H-4: Input Summary Table

Option Number	1	2	3	4	5
Strategy	Status Quo	Redocument	Translate Source Code	Translate & Restructure	Redevelop
Remaining Life Years	12.00	12.00	12.00	12.00	12.00
Investment Years	0.00	0.88	1.17	1.43	1.25
Support Years	12.00	11.12	10.83	10.57	10.75
Reengineered SW O&S	$0	$2,364,344	$2,604,346	$2,265,384	$656,887
Existing SW O&S	$2,571,082	$163,174	$217,411	$264,231	$232,554
Total O&S	$2,571,082	$2,527,518	$2,821,757	$2,529,615	$889,441
Total Benefit	$0	$43,564	($250,675)	$41,467	$1,681,641
Total Investment	$0	$300,621	$694,428	$1,815,506	$1,196,576
Net Value	$0	($257,057)	($945,103)	($1,774,039)	$485,065
Benefit Investment Ratio	N/A	14%	-36%	2%	141%
Total Cost	$2,571,082	$2,828,139	$3,516,185	$4,345,121	$2,086,017

Figure H-5: Economic Assessment Summary Table (Constant Dollars)

H.3 Baseline Estimate (Status Quo)

The system was defined as ground, command & control, and originally documented to Mil-Std 483/490. An annual change rate of 11% was assumed. This is considered nominal for ground systems. No new development; only 12 years of maintenance was estimated. As an aside, the model estimates development and maintenance simultaneously, had there been any development to estimate. Maintenance level (rigor) was set to Hi- because complete maintenance is necessary to keep software from degenerating. An average monthly labor rate of $8,000 per person month was assumed for organic (user performed) maintenance.

SEER-SEM has reports in many shapes and forms which may be analyzed to obtain information. Cost and effort are reported as totals, by month, or by fiscal year in present value (base year) or future value (then year) dollars. Staffing requirements are provided monthly or yearly for maintenance estimates. The estimate of the software support during the redevelopment efforts was performed by taking the steady state maintenance cost per year, found on the *Maintenance Effort By Year* report (see Figure H-6 below), and multiplying it by the number of years that the redevelopment effort is taking place. The report is in base year (constant) dollars. Then year dollars are available in the *Cost By Fiscal Year* report (not reproduced here). The model accounts for increasing complexity over time via code growth.

SEER-SEM (TM) Software Schedule, Cost & Risk Estimation Version 4.51
Project : Software Reengineering Case Stud 9/29/94
CSCI : Maintenance Only of Existing Sys 12:45:10 PM

Maintenance Effort by Year

Base Year: 1994
Fiscal Year Start Month: 10

Fiscal Year	Average Staff Level	Effort Months Correct	Adapt	Perfect	Enhnc	Total	Cum	Base Year Cost	Base Year Cumulative
1994	4.2	18.9	2.3	14.6	2.0	37.8	37.8	302,550	302,550
1995	3.5	17.8	4.4	14.3	5.5	41.9	79.8	335,463	638,013
1996	2.3	5.8	5.8	7.4	8.1	27.2	106.9	217,403	855,416
1997	1.9	3.5	5.7	5.9	8.1	23.2	130.1	185,425	1,040,841
1998	1.9	3.5	5.7	5.9	8.1	23.2	153.3	185,425	1,226,266
1999	1.9	3.5	5.7	5.9	8.1	23.2	176.5	185,425	1,411,691
2000	1.9	3.5	5.7	5.9	8.1	23.2	199.6	185,425	1,597,116
2001	1.9	3.5	5.7	5.9	8.1	23.2	222.8	185,425	1,782,541
2002	1.9	3.5	5.7	5.9	8.1	23.2	246.0	185,425	1,967,966
2003	1.9	3.5	5.7	5.9	8.1	23.2	269.2	185,425	2,153,391
2004	1.9	3.5	5.7	5.9	8.1	23.2	292.4	185,425	2,338,816
2005	1.9	3.5	5.7	5.9	8.1	23.2	315.5	185,425	2,524,241
2006	1.9	0.9	1.4	1.5	2.0	5.9	321.4	46,841	2,571,082

Figure H-6: Maintenance Effort By Year Report

H.4 Redocumentation

Assume no changes to the existing system code; specifications will be redocumented to a tailored DoD 2167A standard. The redocumentation effort was modeled using the SEER-SEM Redocumentation knowledge base. The k-base set the redesign values to a least/likely/most range of 17/29/48% because redesign captures the effort of redocumenting existing code. Reimplementation and retest values were set to 0% because the code is not changing. Maintenance level (rigor) was set to Nom+ because above average maintenance rigor is needed to keep software from degenerating over time. Newly redocumented software makes documentation updates slightly less critical. An average per person monthly labor rate of $8,000 was assumed for organic maintenance, and $14,700 for redocumentation.

H.5 Restructuring (with Language Translation)

Assume no changes to the system's basic functionality; all source code will be restructured and translated to Ada to improve maintainability. Source lines of code for the Assembly and CMS-2 pieces were scaled to equivalent Ada source lines. The scaling factor is based on SEER-SEM's language conversion factors. The restructuring effort was modeled using the SEER-SEM

Minor Reengineering and Automated Language Conversion knowledge bases. These knowledge bases estimate minor rework of an existing application to improve program structure and documentation, along with the rework required to translate the source code to Ada. The k-base also set the redesign, reimplementation, and retest parameter input values. Redesign was set at 17/31/97% because a design change is necessary to translate the existing CMS-2 code to Ada, and restructure to an object oriented paradigm. Reimplementation was set at 8/18/71% because all code will be reimplemented in Ada and restructured to object oriented programming. Retest was set at 72/88/100% because retest is required to verify the CMS-2 to Ada translation and code design modification. Although the solution scenario included the use of an automated language translation tool, an automatic restructuring tool was not considered. These restructuring tools are not available for all languages and are not very robust. An average monthly labor rate of $8,000 per person month was assumed for organic maintenance. The restructuring and translation effort used an average monthly labor rate of $14,700 per person month.

H.6 Translation

Assume no changes to the system's basic functionality; all source code will be rewritten in Ada. Additionally, it was assumed that the CMS-2 portion could use an automated code translation tool. However, no viable translation tool existed for the Assembly portion. The code conversion effort for CMS-2 and Assembly portions was modeled using the SEER-SEM Automated Code Conversion and Manual Code Conversion knowledge bases, respectively. These k-bases assume that most of the recoding effort is accomplished by the tool, however, there will be some manual recoding required. The overall redesign, reimplementation, and retest values were based on SEER-SEM's knowledge bases to 2/6/13%; 3/5/8%; and 44/49/61%, respectively. SLOC for both the CMS-2 portion and the Assembly piece were scaled manually to equivalent Ada source lines before being input to the model. The scaling factor is based on SEER-SEM's language conversion factors. No new code was developed. An average monthly labor rate of $8,000 per person month was assumed for organic maintenance. The translation effort used an average monthly labor rate of $14,700 per person month.

H.7 Retargeting

Although this option was not analyzed for this example, a retargeting effort is easily estimated with SEER-SEM using the Major Rehost or Minor Rehost knowledge bases. These k-bases will capture the effort required to port software from one hardware platform to another.

H.8 Data Reengineering

Since this option was not applicable to the example presented here, it was not analyzed. However, the SEER-SEM Acquisition Method k-bases could easily handle various Data Reengineering projects. The exact choice of k-base would of course depend on the scope and type of changes being made.

H.9 Reverse Engineering

Since this option was not applicable to the example presented here, it was not analyzed. Typically, a reverse engineering effort would be combined with another type of reengineering option such as retargeting and/or restructuring. A reverse engineering effort alone, without any code changes would be estimated by entering 50% redesign with 0% reimplementation and retest against the existing size of the software.

H.10 Reverse Engineering with Other Strategies

As mentioned above, a reverse engineering effort would be combined with another type of reengineering option such as retargeting and/or restructuring. To estimate this, use a combination of Acquisition Method knowledge bases. The redesign, reimplementation, and retest percentages are in most cases additive for the Acquisition Method k-bases. For example, consider the least/likely/most range of three inputs for redesign. The Major Reengineering knowledge base loads 5/10/40%. The Automated Language Conversion k-base loads 2/6/13%. Combining the strategies requires summing the inputs to 7/16/53%.

H.11 Redevelopment

The system will be redeveloped in a fourth generation language using object oriented methodologies. SEER-SEM's New Development knowledge base was used for modeling. The existing sizes were scaled to an equivalent size in a 4GL using SEER-SEM's language conversion factors. In addition, because new developments often invite new requirements, 25 to 50% code growth was assumed. An average monthly labor rate of $8,000 per person month was assumed for organic maintenance. The redocumentation effort used an average monthly labor rate of $14,700 per person month.

H.12 Operations & Support (Before & After Reengineering/Redevelopment)

SEER-SEM's treatment of O&S (maintenance) before reengineering is described thoroughly in section H.3, Baseline Estimate. Operations and support of reengineered software was estimated by inputting a duration for maintenance of the reengineered software. With respect to O&S after reengineering, SEER-SEM automatically schedules the maintenance after the reengineering project.

H.13 Other SEER-SEM Features Used To Support Re-Engineering Economic Analysis

SEER-SEM is a tool designed to provide the analyst with a flexible environment in which to analyze various scenarios, contingencies, and configurations. There are many features which may be used to assist analysis of various reengineering options. Some specific features include risk analysis, sensitivity analysis, and documentation via input notes.

Risk Analysis

SEER-SEM permits the entry of ranges for parameters which allow the estimator to capture the risk and uncertainty of a reengineering effort into the cost estimate. In addition, cost ranges for various levels of probability of completion may be examined. For example, the most likely effort for the redevelopment strategy was estimated to be 81.40 effort months (see Figure H-7 below), however, the ranges expressed in the size and other input parameters really mean there is a range about this most likely estimate. SEER-SEM has a risk analysis report and several charts to show the range on estimated cost, effort, schedule, or defects. Staffing levels also may be assessed. The Staffing Plan screen capture (see Figure H-8) illustrates staff levels over time for both requirements and development activity phases.

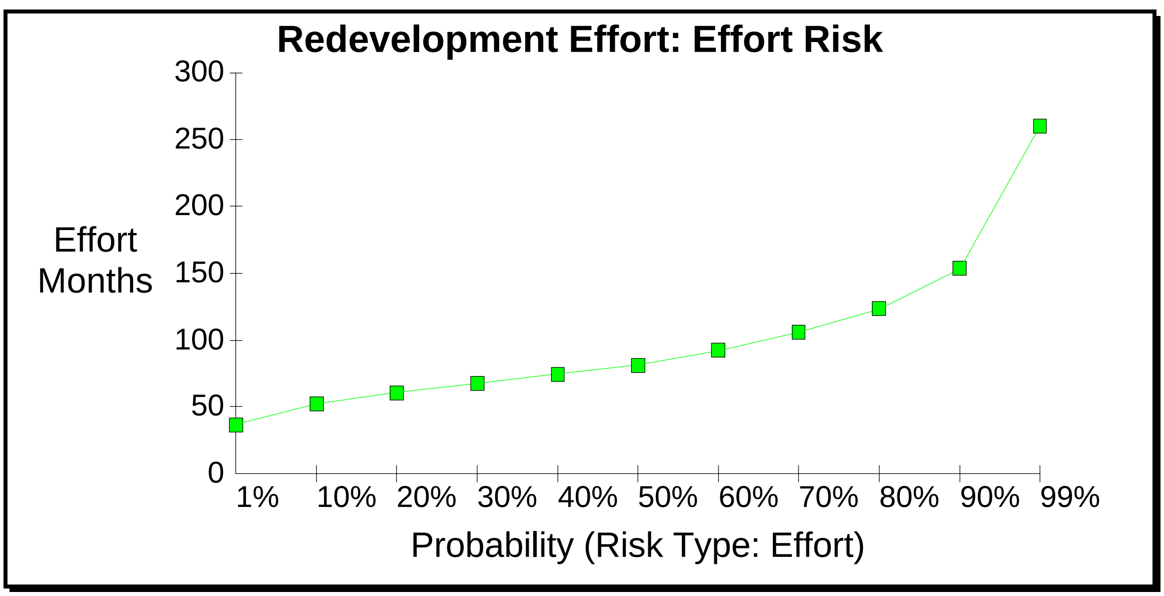


Figure H-7: Effort Risk Analysis for Redevelopment Effort

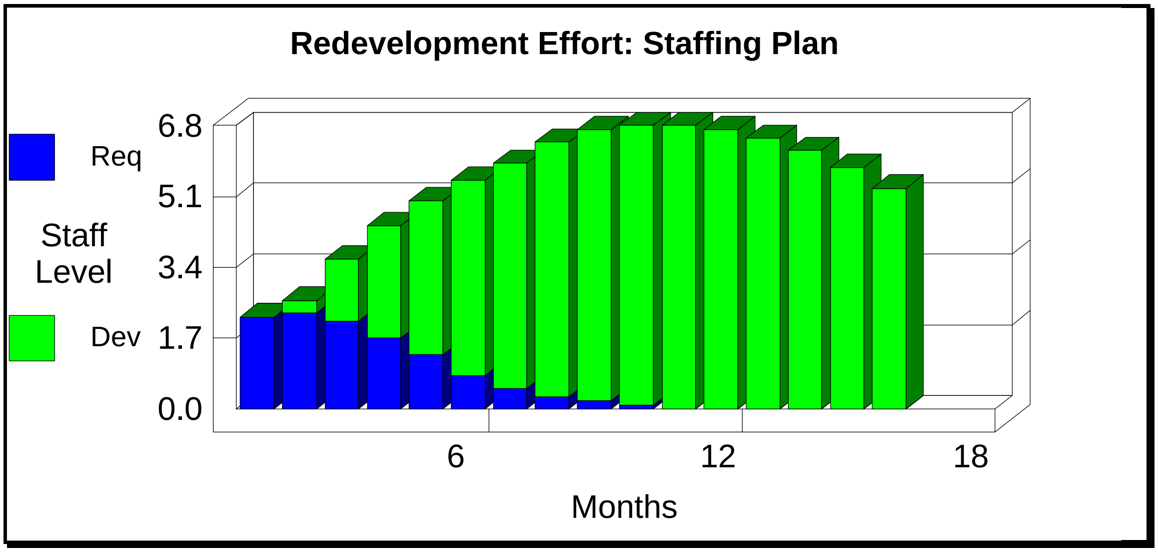
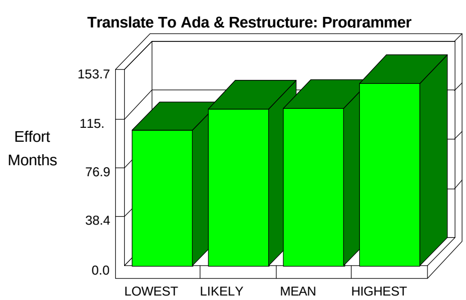


Figure H-8: Staffing Plan For the Redevelopment Effort Shows Staffing Levels Over Time

Sensitivity Analysis

SEER-SEM supports sensitivity analysis with hundreds of charts depicting the cost, schedule, effort, and defect ranges for any parameter (see Figure H-9 below). In fact, the entire range of a parameter's sensitivity may be plotted for more detail. This is of particular value in an economic assessment of various reengineering options. For example, you may be looking at the possibility of translating existing code to Ada, or restructuring the code to an object oriented format. The capabilities and experience of the programming team will have a direct bearing on the economic viability of these options. Knowing how your estimate could swing as a result of programmer's capabilities, or any other parameter for that matter, is valuable information.



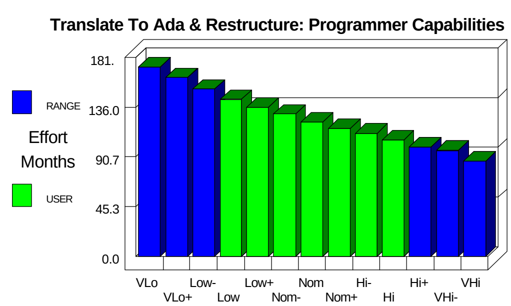


Figure H-9: Sensitivity of Programmer Capabilities - User Range vs. Full Parameter Range

Input Notes

SEER-SEM has the ability to track the rationale and/or source for each input parameter, as well as overall ground rules and assumptions. Effective use of input notes creates an audit trail that pays dividends when a project must be revisited months after the original estimate. The marginal cost in time to document the project file with input notes is far outweighed by the marginal benefit of time savings when the notes are needed to understand parameter settings. For this case study, notes were entered for each parameter that was changed from the knowledge base default. All notes can be viewed on the *Inputs with Notes* report (see Figure H-10). The line item "Pre-exists, not designed for reuse" is an output calculated from pre-existing LOC and parameters input by the user. Other features such as clipboard support for all reports and charts and exporting to a spreadsheet also assist the documention of an estimate.

Project : Software Reengineering Case Study 9/29/94
CSC : CMS-2 Converted to Ada 1:25:19 PM

Inputs with Notes

	Least	Likely	Most	
+ LINES				
- New Lines of Code	0	0	0	
+ Pre-exists, not designed for reus	5,594	6,903	9,539	
- Pre-existing lines of code	34,007	34,007	34,007	Source lines of code for the CMS-2 piece scaled to equivalent Ada source lines. Scaling factor base on SEER-SEM's language conversion factors.
- Lines to be deleted in pre-exst	0	0	0	
- Redesign required	2.00%	6.00%	13.00%	Design change required due to changing from CMS-2 to Ada. Values based on SEER-SEM's automated code translation knowledge base.
- Reimplementation required	1.00%	3.00%	6.00%	All code will be reimplemented in Ada. Values based on SEER-SEM's automated code translation knowledge base.
- Retest required	44.00%	49.00%	61.00%	Retest required due to changing from CMS-2 to Ada. Values based on SEER-SEM's automated code translation knowledge base.

Figure H-10: A Portion of the Inputs With Notes Report

APPENDIX I

SLIM SOFTWARE ESTIMATING MODEL

This appendix describes how the SLIM software estimating methodology can be applied to reengineering efforts. It was prepared by Mr. Larry Putnam, Jr., and Mr. Doug Putnam of Quantitative Software Management and edited by Mr. John Clark of Comptek Federal Systems.

Mr. Larry Putnam, Jr.
Mr. Doug Putnam
Quantitative Software Management, Inc.
2000 Corporate Ridge, Suite 900
McLean, VA 22102
(703) 790-0055 (Voice)
(703) 749-3795 (Fax)
76274.72@compuserve.com

The SLIM (Software Life-cycle Management) methodology was initially formulated into a commercial product offering by Lawrence H. Putnam in 1978 after 10 years of pioneering research into software production. The methodology has been continuously refined and enhanced to ensure that it is current and capable of supporting the ways that organizations are building software products today.

INPUTS:

SLIM requires three primary inputs. The first input is the proposed size of the application. SLIM is flexible enough so that any of the popular sizing metrics can be used. These are a few of the sizing metrics that could be used.

- Source Lines of Code
- Function Points
- Objects
- Windows
- Screens
- Diagrams

SLIM uses a range estimating approach on the size inputs. Size estimating can be one of the most difficult areas to quantify (no four decimal places of accuracy). To cope with the inherent uncertainties early in a project when the least about the size of a project is known we input 99% ranges around our best guess for the expected size. SLIM uses this +/- σ information to determine the probability of success (more on this in the discussion of dynamic risk assessment).

The second major input section is productivity and complexity. There are three levels of detailed information that can be entered into SLIM. The level of detail depends on how much is known about the development team, types of tools & methods to be used, and the application complexity. Users can input their known productivity based on historic project calibrations or they can have SLIM determine an appropriate productivity level based on answers to the detailed questions (more on this in project assumptions).

The final input area is one that is often left out of other tools. They are the management constraints. These include the following:

- The desired schedule
- The desired budget limit
- The desired reliability (Acceptable Mean Time To Defect) at delivery
- The minimum staffing required to have the skill mix to get the job done
- The maximum practical staffing possible

Figure I-1 below shows the high level conceptual model for SLIM. The user supplies the project assumptions and goals. The tool can be customized to the development organizations lifecycle methodology through the user customized settings. The developer organization's historic data is entered into SLIM both as a basis for generating future estimates and for validating those estimates. When all possible planning alternatives have been explored there is an automated presentation capability.

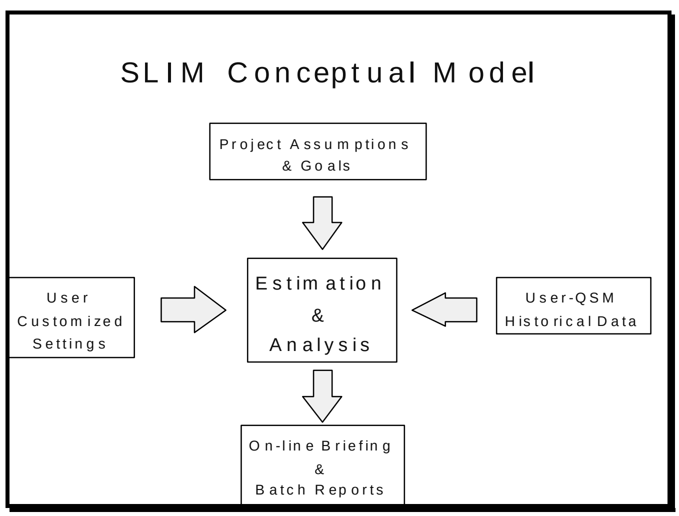


Figure I-1. Conceptual Model of the SLIM Estimation Tool

PROCESS:

SLIM uses this input information to determine an "optimum" estimate. The "optimum" estimate is a solution that gives the highest probability of developing the system within the management constraints that have been specified. If the constraints are too tight then the "optimum" estimate will exceed one or a number of the goals. If this is the case one must evaluate other practical alternatives. These might include scenarios for reduced function products, increased staffing or improved efficiency. Variations of the basic estimate can be logged so that they can be compared to the merits of each alternative and make a decision about which estimate is the best.

OUTPUTS:

There are 181 different reports and graphs that are available in SLIM. The outputs are grouped into the following major categories:

- Project Description
- Estimation Analysis Views
- Schedule Section
- Risk Analysis Section
- Staffing & Skill Breakout Section
- Effort and Cost Section
- Reliability Estimate Section
- Documentation Section

An outline editor allows creation and storage of briefing charts so the estimator can quickly and efficiently communicate the results of their estimates to the key decision makers (internal or customers).

SLIM for Windows Architecture

Figure I-2 below shows the basic SLIM architecture.

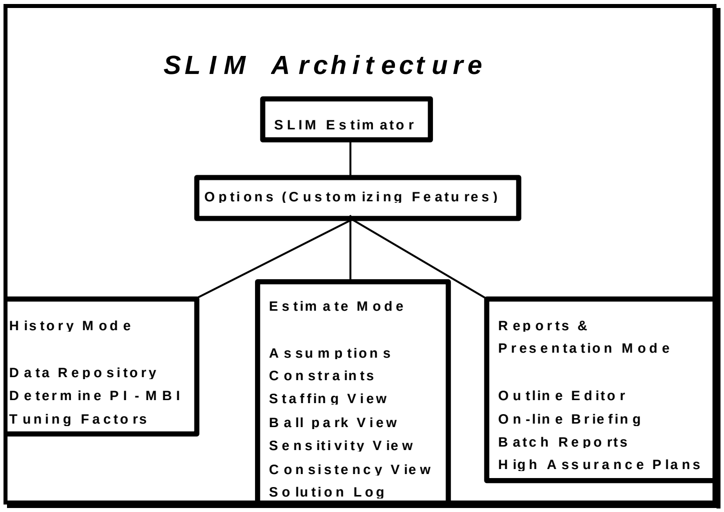


Figure I-2. SLIM for Windows Architecture

The three major modes of operation in SLIM:

- History Mode
- Estimation Mode
- Reports Presentation

In History Mode the user inputs historical data and determines some key parameters. Figure I-3 shows an example of the data for a completed project. The projects are calibrated and used to customize the SLIM tool to their development environment.

The Estimation Mode provides the user with several estimation views. Each view is designed to let the user easily analyze a particular situation in order to arrive at a good solution in the most efficient manner possible. In Estimation Mode the historical data can be used to validate a new estimate.

The Reports and Presentation Mode provide the user with an effective capability to create and deliver on-line briefings and reports.

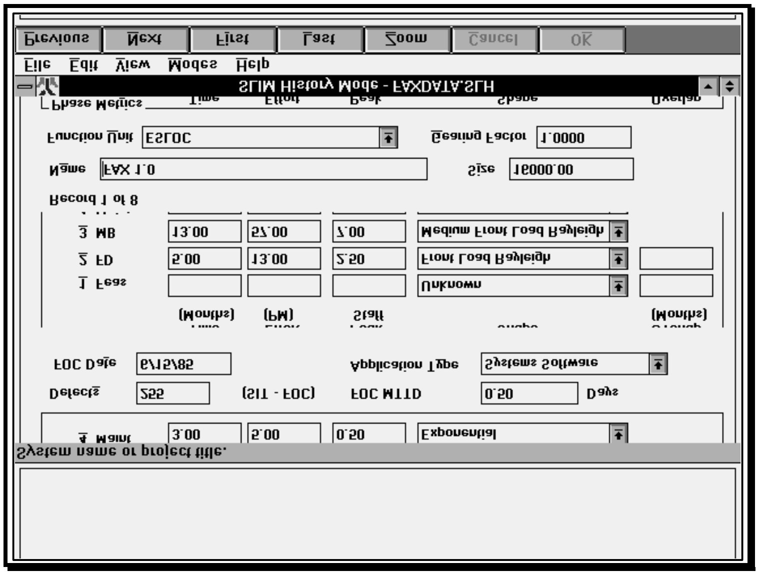


Figure I-3. Inputs for Historically Completed Projects

SLIM History Mode—Calibrating the Tool to the Way Business is Done

The purpose of the history mode is to provide the user with a way to capture historical data. SLIM only requires the core metrics recommended by the Software Engineering Institute (Size, time, effort and defects). The tool will then run the data back through its estimation equations to calculate each projects "Productivity Index" (a macro measure of efficiency that ranges from 1-40. A higher number indicates a more productive process) along with some other fine tuning parameters.

SLIM Estimation Mode—The Heart of the System

The estimation mode is where users will spend most of their time. It contains the project estimation assumptions, project constraints, and all of the analytical views.

Assumptions:

There are three levels of detail that a user can choose from as inputs for the estimate. At the top level SLIM will only require one screen of information. The inputs are shown in Figure I-4. The top level input is most appropriate for the users that have sufficient historical data and can select a reasonable value for the Productivity Index. Another reason may simply be that the user has a preference toward estimating from a high level perspective.

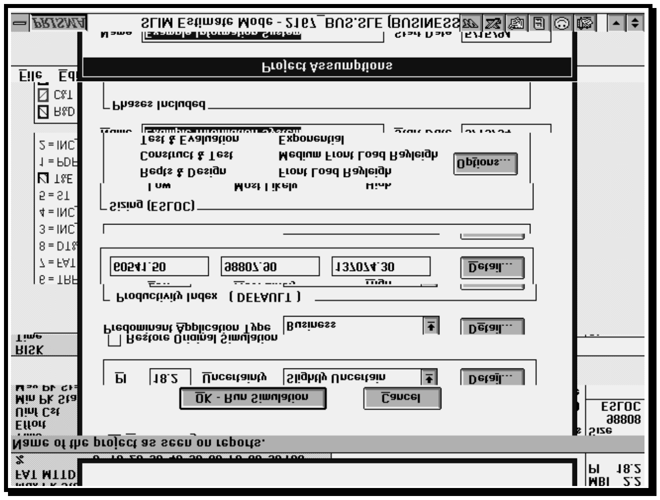


Figure I-4. Minimum Inputs Required to Run an Estimate with SLIM

For those users that prefer a greater level of detail or would like SLIM to determine a Productivity Index for them, SLIM offers a second level of detail. This is shown in Figure I-5. At the second level, the user inputs a value from 0 to 10 for three major productivity influencing categories. The three categories are:

- Tools & Methods Capability
- Technical Complexity
- Personnel Profile

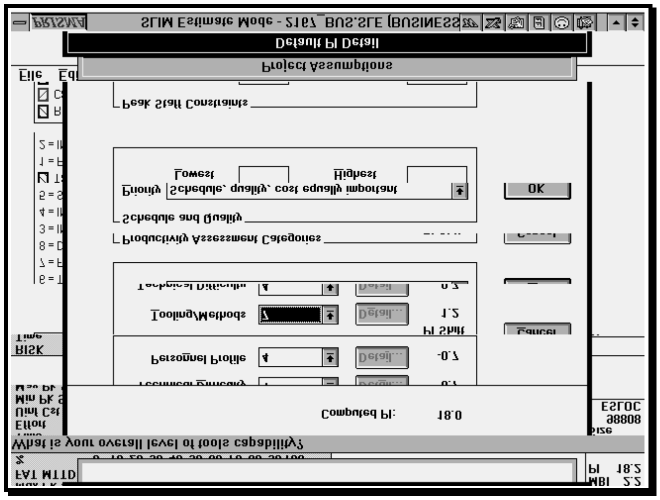


Figure I-5. Level Two Details Inputs for Productivity Index Determination

With this information SLIM goes into its historic data base of over 3,800 projects. It finds a subset of projects with the same size and application complexity. The tool determines the average productivity profile. Finally, it makes adjustments to the average value based on the users' input to the productivity influencing categories.

A third level of detail is available for the users that have a very good understanding of their development environment, or simply prefer a more detailed approach to estimation. Each of the three productivity categories has a detailed series of questions. Figure I-6 shows the detailed screen for the tools and method's category. The user can fill in as many of the inputs as they like and SLIM will use these inputs to determine the Productivity Index.

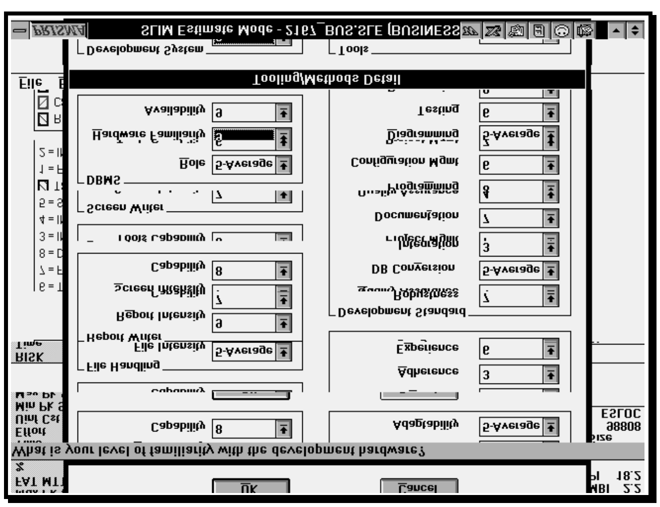


Figure I-6. Detail Inputs for Tools and Methods Capability

The user can determine which level of detail they prefer. In fact, they can even use a combination. For example, the estimator might have a very good understanding of the tools and methods category and choose to enter information at the detailed level. It might not be known who will be assigned to this project so this category could be answered at the second level of detail.

Management Constraints

The final sets of inputs are the management constraints. Some people think or these as the project goals. They are really the boundaries that this project must live within for it to be considered successful at its completion.

SLIM asks the estimator to identify any of six potential management constraints. These include the following. Which are also shown in Figure I-7.

- The desired schedule
- The desired budget limit
- The desired reliability (Acceptable Mean Time To Defect) at delivery
- The minimum staffing required to have the skill mix to get the job done
- The maximum practical staffing possible

Additionally, SLIM would like to know the desired probability of not exceeding the goal and which goal is most important, second most important, etc. If there are no constraints then it is not necessary to fill in this dialogue box.

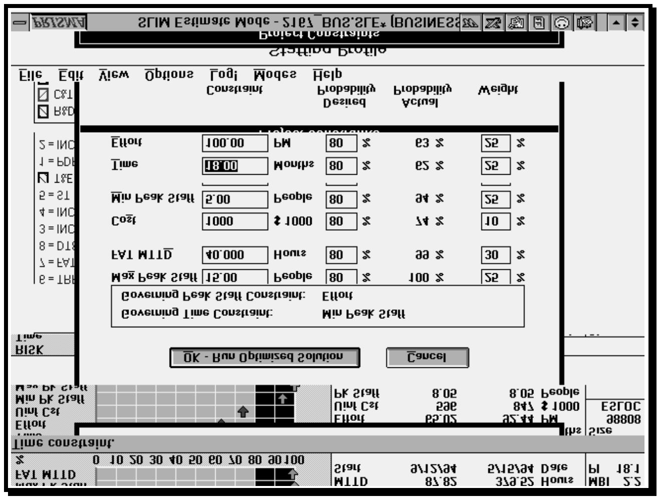


Figure I-7. Project Constraints and the Users Desired Probability of Not Exceeding the Constraint

Optimum Estimate:

Once this information is in the system SLIM will do an analysis to determine the "optimum" estimate. It does this by examining a number of potential solutions from the shortest possible schedule estimate to the longest. For each possible estimate SLIM calculates a joint probability of success. It determines which solution has the highest probability of success and it posts that estimate to the Staffing View.

Staffing View & Dynamic Risk Analysis

The Staffing View is one of five Estimation Views. The purpose is to allow the user to:

- Estimate directly with a resource profile
- Evaluate alternative plans by shifting phases, changing the peak staffing and changing staff loading profiles
- Dynamically see the probability of an estimate meeting the project constraints
- Evaluate alternatives by changing size and productivity assumptions

Figure I-8 contains most of the relevant features of this view. In the middle of the screen is the staffing plan for different phases of the project. There are two handles that are associated with each phase. Handles are used to graphically manipulate the estimate on the screen. The right most handle is the Staffing Schedule handle. It allows the estimator to compress or elongate the schedule by adding or decreasing staff. The user simply moves the mouse over the handle then he clicks and drags the handle to a new position. To compress the schedule move the handle to the right. When the user drops the handle the new staffing plan is on the screen and the new estimate numbers are updated in the solution panel.

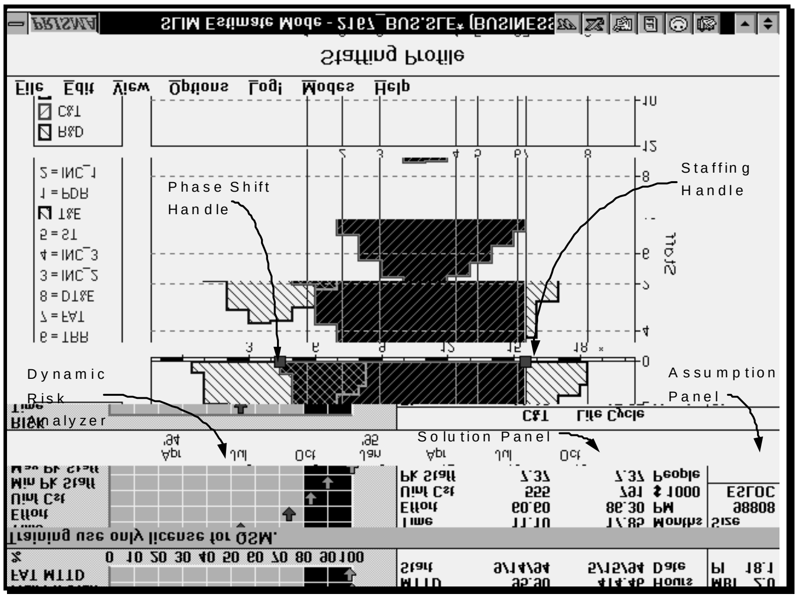


Figure I-8. SLIM Staffing View

While the schedule manipulation is happening the Dynamic Risk Analyzer is being updated. The arrows dynamically move and show the new probability of being able to achieve each of the six project goals. Figure I-9 shows the key features of the Dynamic Risk Analyzer. We find that managers like this view because they can clearly and rapidly see the relationships between cost, schedule reliability and risk as they try different "what if" scenarios.

Figure Not Provided

Figure 1-9. Dynamic Risk Analyzer

Ballpark View - Validating Estimates With History

Ballpark view allows you to graphically compare your estimate to your historic data. This is most useful if:

- A customer is asking you to do the impossible
- You need an independent validity check
- You want to graphically do sensitivity analysis

Figure I-10 shows key features of the Ballpark View. Each of the hollow squares is a historical project. The solid square is the current estimate. This is to make sure that the estimate is reasonably consistent with what has been accomplished in the past. The estimates can be graphically manipulated similar to the Staffing View by click and drag actions on any of the handles. These include Size, Schedule, Effort, or Productivity Index. To see the staffing implications of a change you simply switch back the Staffing View. This is a highly interactive tool.

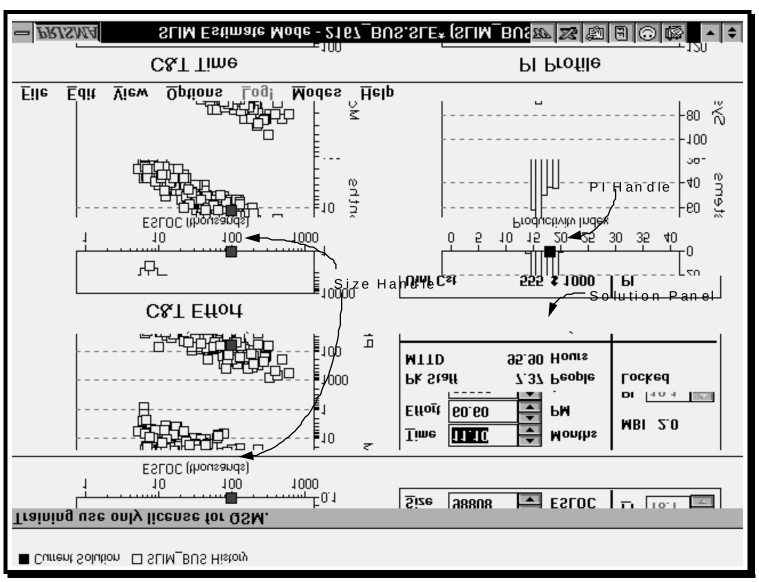


Figure I-10. Ballpark View. This Shows an Estimate Consistent with History

Figure I-11 shows an estimate that conforms to a customer desired schedule and budget. Note that it is below what has ever occurred historically for both schedule and effort. There is a very low probability that the system could be built to meet this set of conditions.

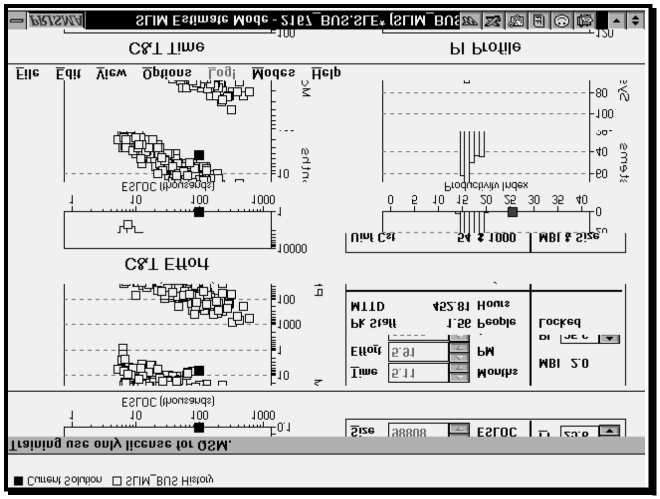


Figure I-11. Unrealistic Estimate Compared to Historical Data

The Ballpark View adds a measure of credibility to your estimates. We find that a factual portrayal of the estimate and history will help you avoid emotional arguments with customers. You can then focus on practical alternatives (like a phased release scenario, etc.).

I.2 Model Inputs and Sample Case Study Description

The sample case study involves different reengineering scenarios for a Management Information System (MIS) developed for the Air Traffic Control system. The expected remaining life of the system is 12 years. The current system has 31,574 source instructions. The system is primarily implemented in CMS-2 (28,417 source instructions) and the remainder is implemented in Assembly (3,157 source instructions). The current system is being maintained at a cost of more than $250,000 a year. The system is a candidate for reengineering and 5 possible strategies are being considered. The strategies to be evaluated with SLIM are:

- maintaining the existing system as it currently is implemented
- redocumenting the existing system
- translating the existing source code to Ada
- restructuring the source code in CMS-2
- completely redeveloping the system in Ada

Each of these cases was analyzed with SLIM and is summarized in the following sections.

There are three primary inputs that need to be provided to SLIM in order to produce an estimate. These inputs are the size for the system to be estimated, the productivity of the development organization and the application type of the product (MIS, Telecom, Realtime, etc.). In the following examples the size was determined by taking the existing system (31,574 CMS-2 and Assembly source statements) and making appropriate adjustments for the development language (Ada). The application type was given in the case description as being MIS. In SLIM this type of application is interpreted as a **business** application.

The productivity of the development organization is determined in SLIM by calibrating to some of organization's own historic data. When we do this we can make an appropriate choice of what Productivity Index (PI) to use for estimating a system to be developed by that organization. In this reengineering example there was no historic data mentioned. In cases where there is no historic data there are two methods for determining a PI. One method is to describe the project in three broad categories (Tooling and Methods, Technical Complexity, and Personnel Profile) and let SLIM select one based on the application type and project description. The other method involves using the QSM database of over 3,800 projects to select a subset of representative projects and calibrating to those projects. This is the method used for estimating the above cases.

1.3 Status Quo

SLIM was used to determine the support costs for maintaining the status quo. A productivity index of 12.4 was selected for this case. This PI was determined from comparing PI trends back eight years to when the system was originally developed. An estimate was made with SLIM of what the original development and associated maintenance costs would be for the system. These support costs were then extrapolated over the remaining twelve year life of the system.

1.4 Redeveloping the System

Figures I-12 and I-13 below show the project assumption input to SLIM. A brief description follows explaining the inputs.

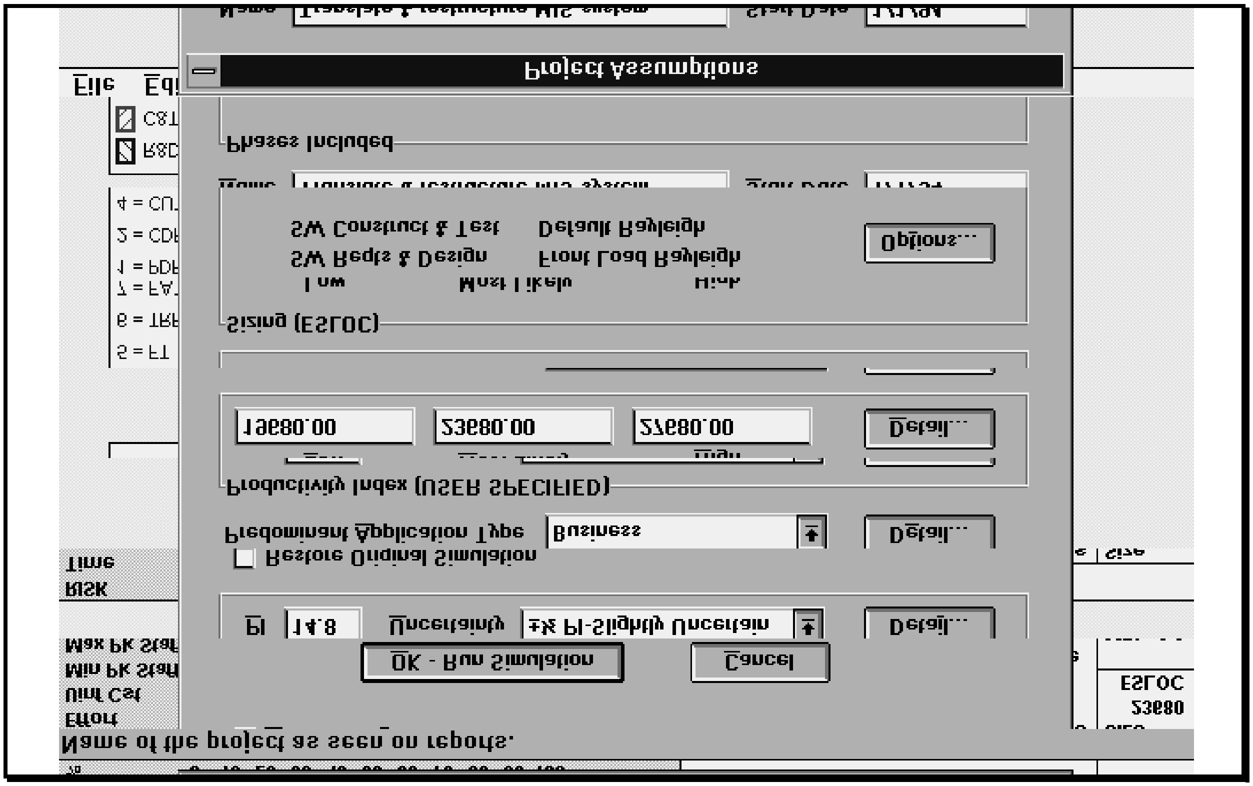


Figure I-12. Redevelopment Case SLIM Estimate Inputs

The start date for the project was arbitrarily chosen as January 1, 1994. The phases that were included were Software Requirements & Design and Construction & Test. SLIM would expect that 95% of the software defects would be removed at the end of the Construction & Test phase. SLIM provides the capability to customize the staff loading profiles. Since no guidance was given in the case example, SLIM picked a default staffing template.

SLIM uses a three point size range for estimating. The estimator provides the distribution and SLIM calculates the expected size and the associated uncertainty (standard error of the estimate). This uncertainty in the size is then mapped into uncertainty in the schedule and cost using a Monte Carlo simulation process. This becomes important when we are assessing the risk of meeting a specific date and budget figure. In this case the size was determined manually by applying an average gearing factor for lines of code per function point for CMS-2 and Assembly to determine the expected number of function points. Then a gearing factor for Ada was applied to convert the function points into an appropriate number of Ada statements. An escalation factor was also used to account for new functionality that is typically accommodated when a system is redeveloped.

The Productivity Index (PI) was determined by using QSM's PADS metrics repository to select an appropriate set of projects that matched the redevelopment case. The criterion for selection of the projects was redeveloped business systems delivered since 1990. The sample that matched that criteria consisted of 56 projects. The PI used in this estimate was close to the average for the data set. If this were a real estimate the development organization's historic data would be substituted for the QSM data set.

The PI uncertainty in this case was set to slightly uncertain. The uncertainty on the PI has an impact when the simulation is run. More uncertainty in the PI has an effect when SLIM calculates the risk of making a specified delivery date and budget figure.

Although it was not specified in the case example, SLIM has the ability to determine an optimum plan given a set of project constraints. Below are some project constraints that were arbitrarily determined for this project.

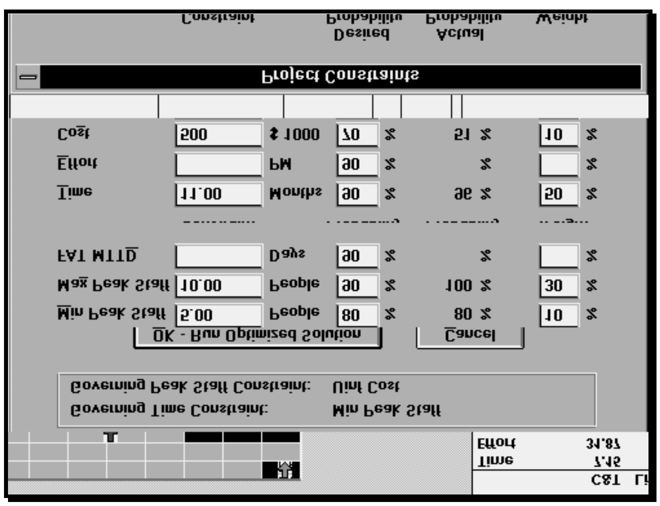


Figure I-13. Redevelopment Case Sample Constraints

The estimator would input constraints that he or she was concerned about and their desired probability that they would not exceed those constraints. The estimator would then rank the constraints by providing a percentage weight. SLIM will then use this information to determine an optimized plan with respect to the above constraints.

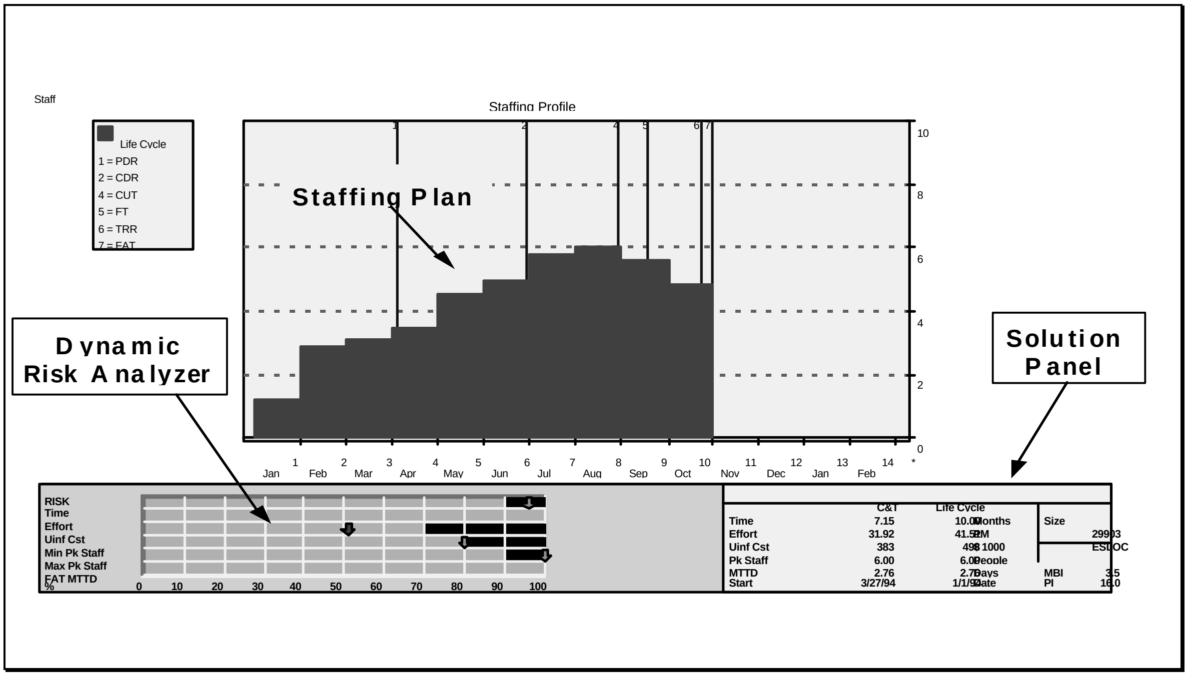


Figure I-14 is the Staffing View of the Redevelopment Estimate in SLIM.

The Staffing View is divided into three areas. The center of the screen is the monthly staffing estimate. The numbered lines reflect the major milestone events. The milestones are customizable and in this example are setup for 2167A. The lower right portion of the screen shows the solution panel. This is where the schedule, cost, staffing and quality information is summarized. The lower left portion of the screen shows the risk panel. The risk gauge shows the probability that this plan will satisfy the constraints and desired probabilities entered earlier. The arrows represent the actual probabilities of the current plan. The dark shaded portion of the risk gauge is the estimator’s desired probability. If the arrow is in the desired region it turns green indicating that the current plan satisfies the estimators risk criteria. If the arrow is outside the desired region it turns red indicating that the probability of meeting the constraints has not been satisfied.

SLIM has multiple estimation views that are useful for displaying different information. The Ballpark View is useful when you are want to compare your estimate with your historic performance to make sure that the estimate is achievable. Figure I-14 is the Ballpark View screen from SLIM for the Redevelopment Case.

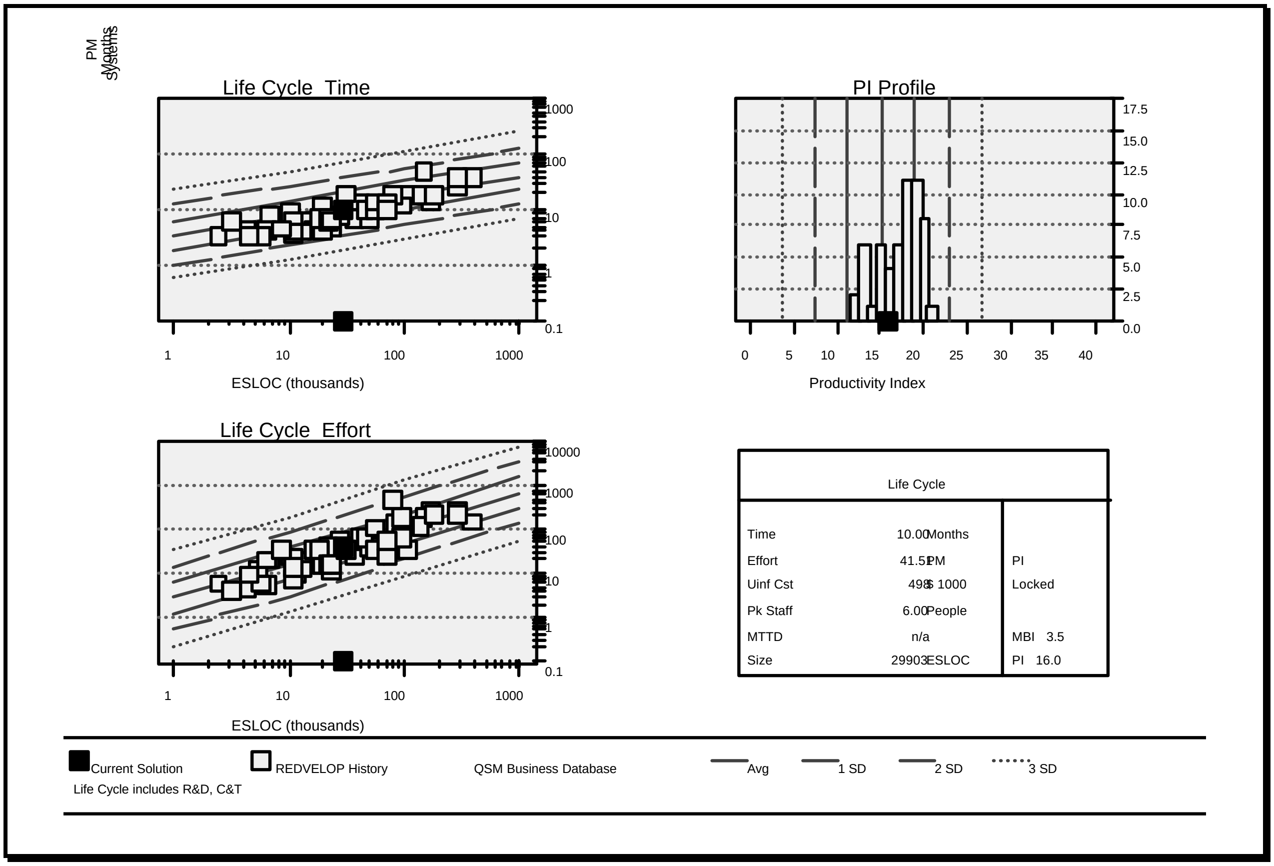


Figure I-15. Redevelopment Ballpark View Estimate Compared with 56 Historic Projects From the QSM Database

There are three graphs on the Ballpark screen, Schedule vs. Size, Effort vs. Size and a Productivity Index histogram. In the Ballpark screen above the SLIM estimate for the redevelopment case is portrayed as the black square. The white boxes and bars show the 56 comparative projects from the QSM Database. The lines on the schedule and effort graphs represent trendlines fit to the overall current QSM business system database. The thick middle line is the average fit and the others represent 1,2 & 3 standard deviation lines. Notice that the estimate for the redevelopment case fits well within the historic data distribution on the schedule and effort graphs. This indicates that the historic data confirms that the plan is achievable. This is further reinforced by looking at the PI profile graph. The PI for our plan is in the middle of the PI distribution of the historic projects again indicating that the plan is consistent with historic performance. If this were a real case the data on these graphs would come from the development organization.

SLIM provides a rich array of graphic output. In the Reports and Presentation Mode there is even a slide show capability to help communicate the estimate to different audiences. On pages I-16 through I-18 are some examples.

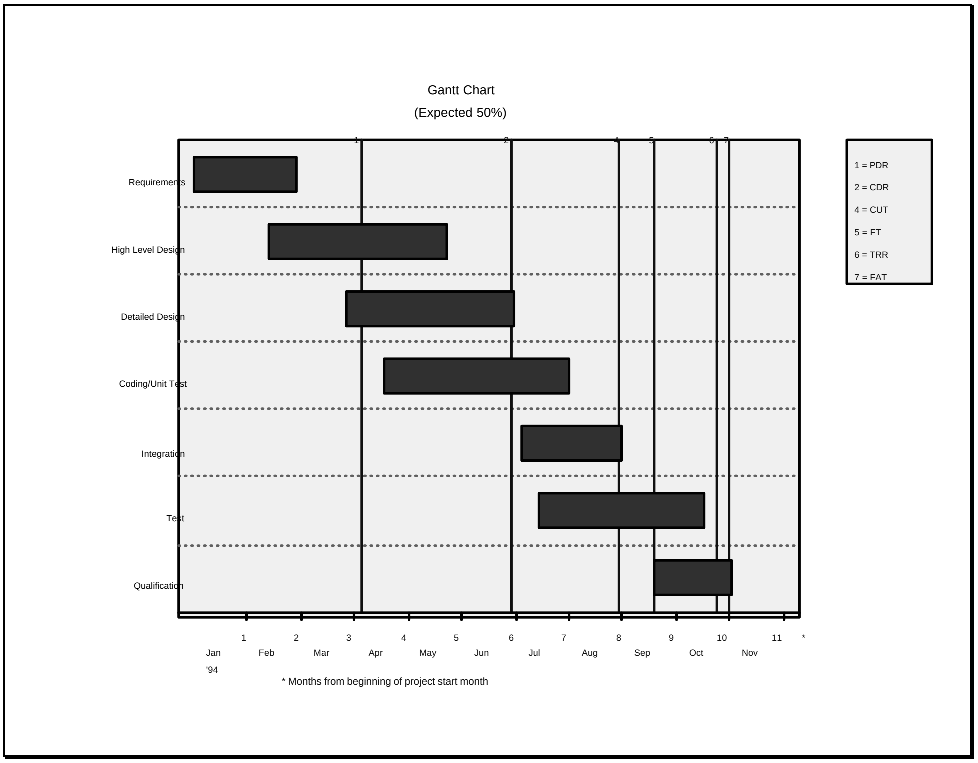


Figure I-16. Gantt Chart From SLIM

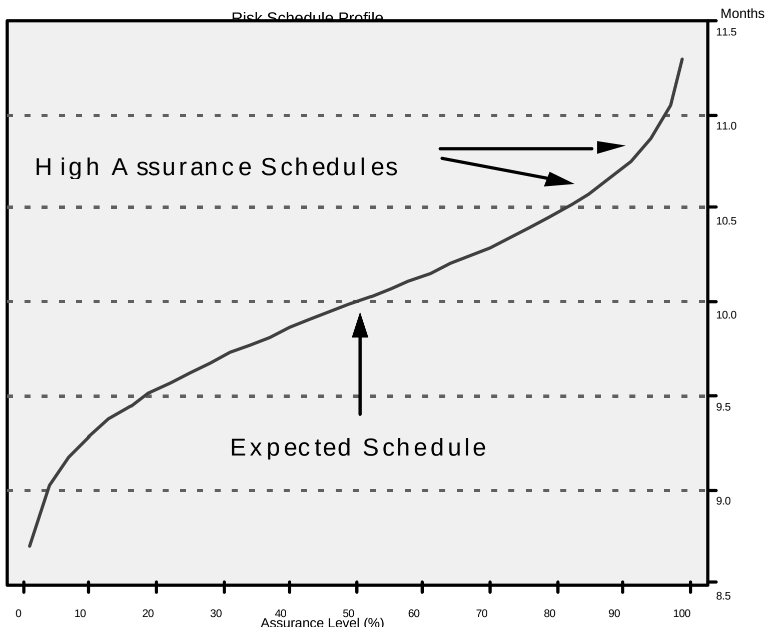


Figure 17. Schedule Risk Graph From SLIM

Figure Not Provided

Figure I-18. Cumulative Cost by Phase Graph From SLIM

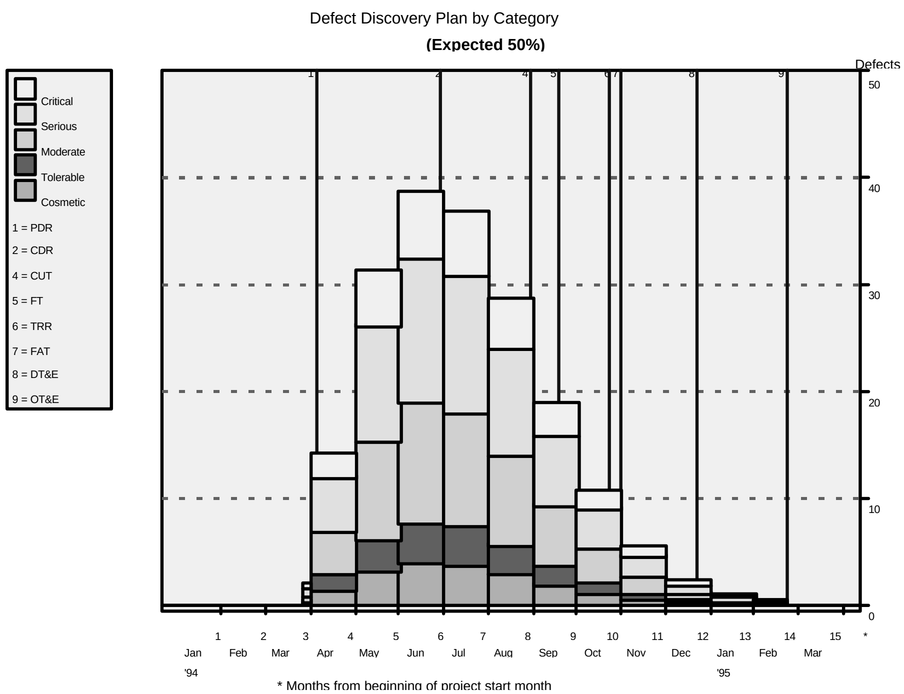


Figure I-19. Defect Rate by Category Forecast From SLIM

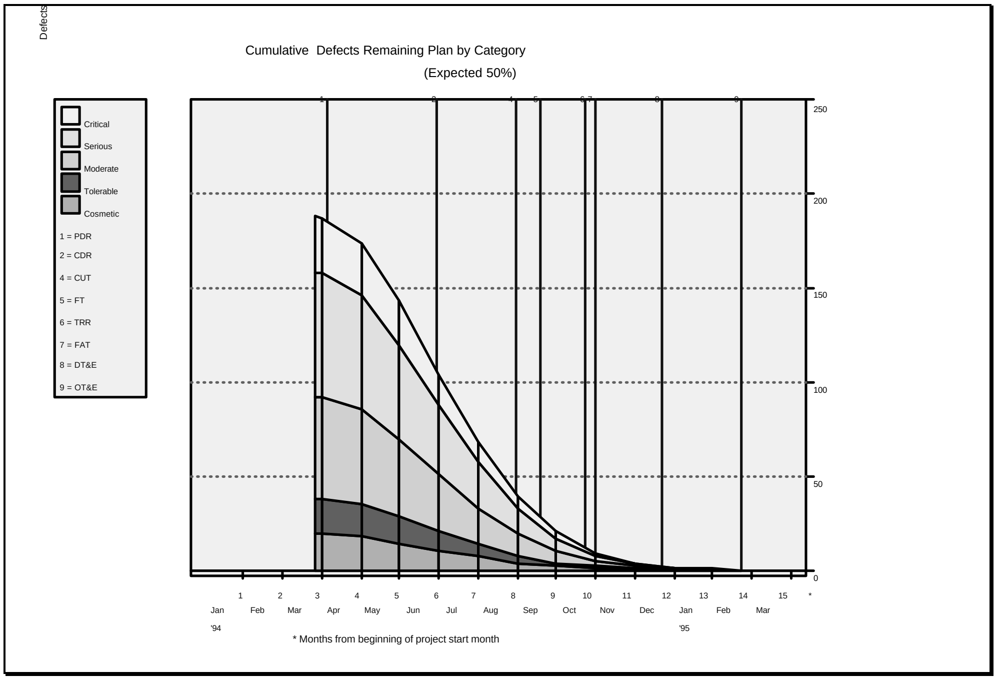


Figure I-20. Defects Remaining Forecast From SLIM

SLIM handles support costs by modeling each release of the system. The inputs needed to generate the support costs are similar to those entered for the redevelopment case earlier. Usually the sizing is more involved because adding new functionality requires changes to existing functionality. Because the case example was set up to model twelve years of support costs, SLIM was used to determine the effort expenditure for twelve months of support. This figure was used to determine the remaining lifecyle support costs.

1.5 Restructure

Figure I-21 below shows the input for the restructuring case. A brief summary of the SLIM input parameters is provided below.

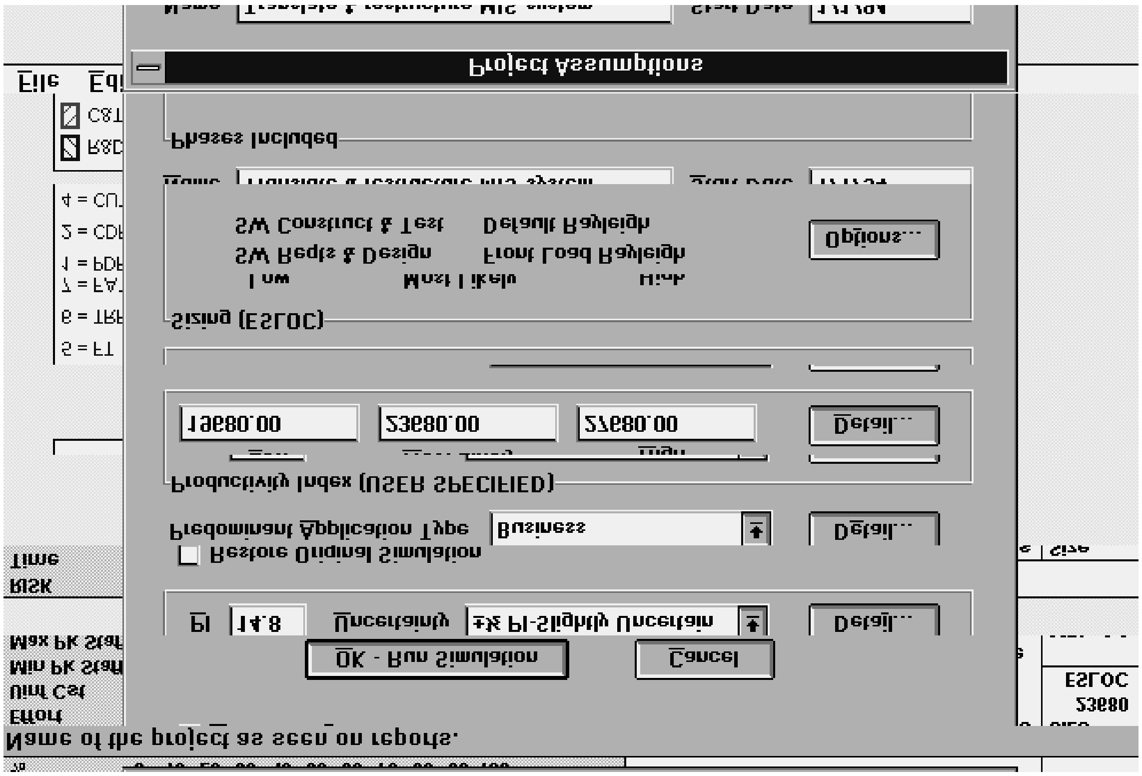


Figure I-21. Estimate Assumptions For Restructure Case

All parameters remained the same as the redevelopment case except the size and the Productivity Index. The size was determined by assuming that 75% of the existing source lines of code would have to be reworked during restructuring.

The Productivity Index was determined by pulling out a set of projects from the QSM Database that match the characteristics of this case. The criteria for selection were business systems that were restructured since 1990. There were 202 projects that matched these criteria. The PI used in this example was an average for the data set.

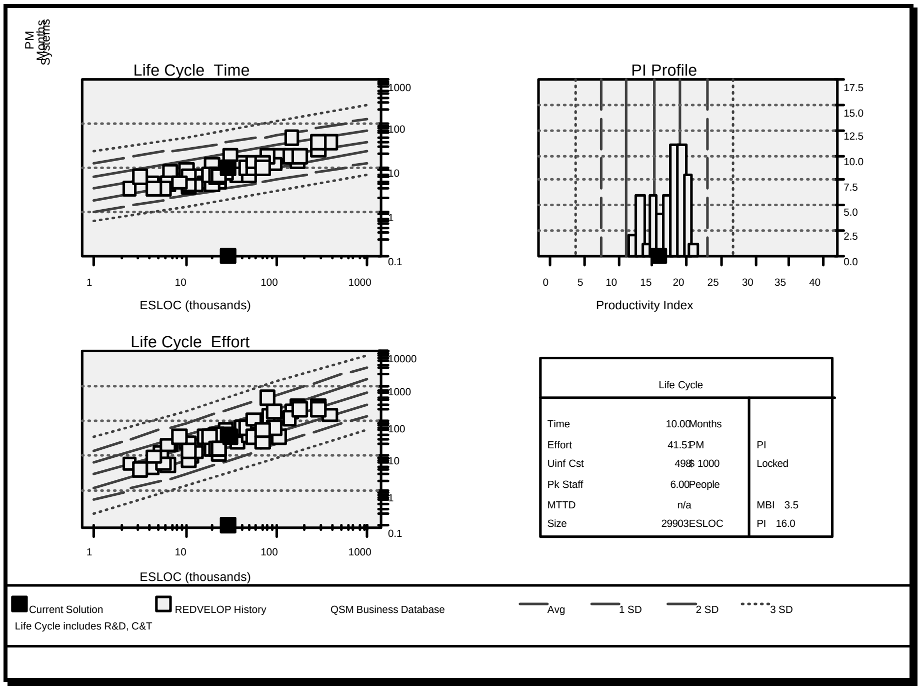


Figure I-22. Restructure Estimate Compared to Historic Data in the Ballpark View in SLIM

Notice that the estimate for the restructure case fits well within the historic data distribution on the schedule and effort graphs. This indicates that the historic data confirms that the plan is achievable. This is further reinforced by looking at the PI profile graph. The PI for our plan is in the middle of the PI distribution of the historic projects again indicating that the plan is consistent with historic performance. If this were a real case the data on these graphs would come from the development organization.

The support costs were calculated using the same method as the Redevelopment Case.

1.6 Translation

Figure I-23 shows the input for the translation case. A brief summary of the SLIM input parameters is provided below.

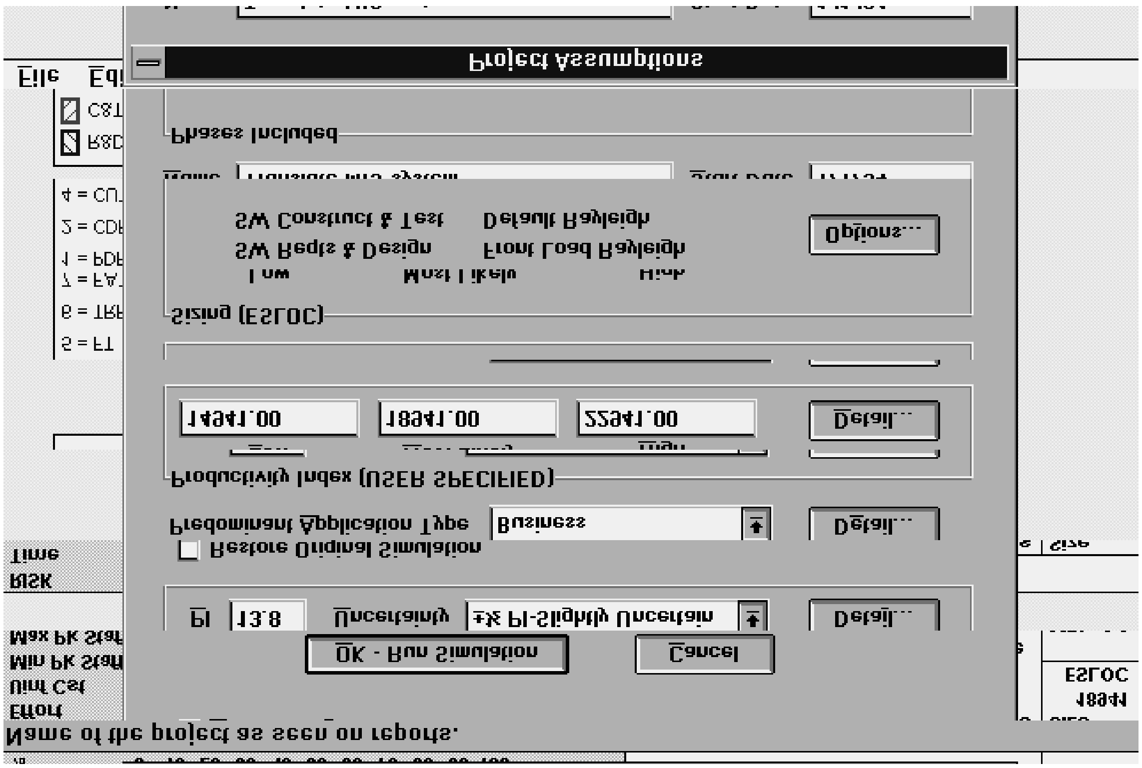


Figure I-23. Estimate Assumptions for Translation Case

All parameters remained the same as the Redevelopment Case except the size and the Productivity Index. The size was determined as before using a function point language gearing factor for CMS-2 and Assembly to map the existing source lines of code into function points. Those function points were then mapped into Ada source lines of code by applying a gearing factor appropriate for Ada.

The productivity index was determined by pulling out a set of projects from the QSM Database that match the characteristics of this case. The criteria for selection were business systems that were translated since 1990. There were 22 projects that matched these criteria. The PI used in this example was an average for the data set.

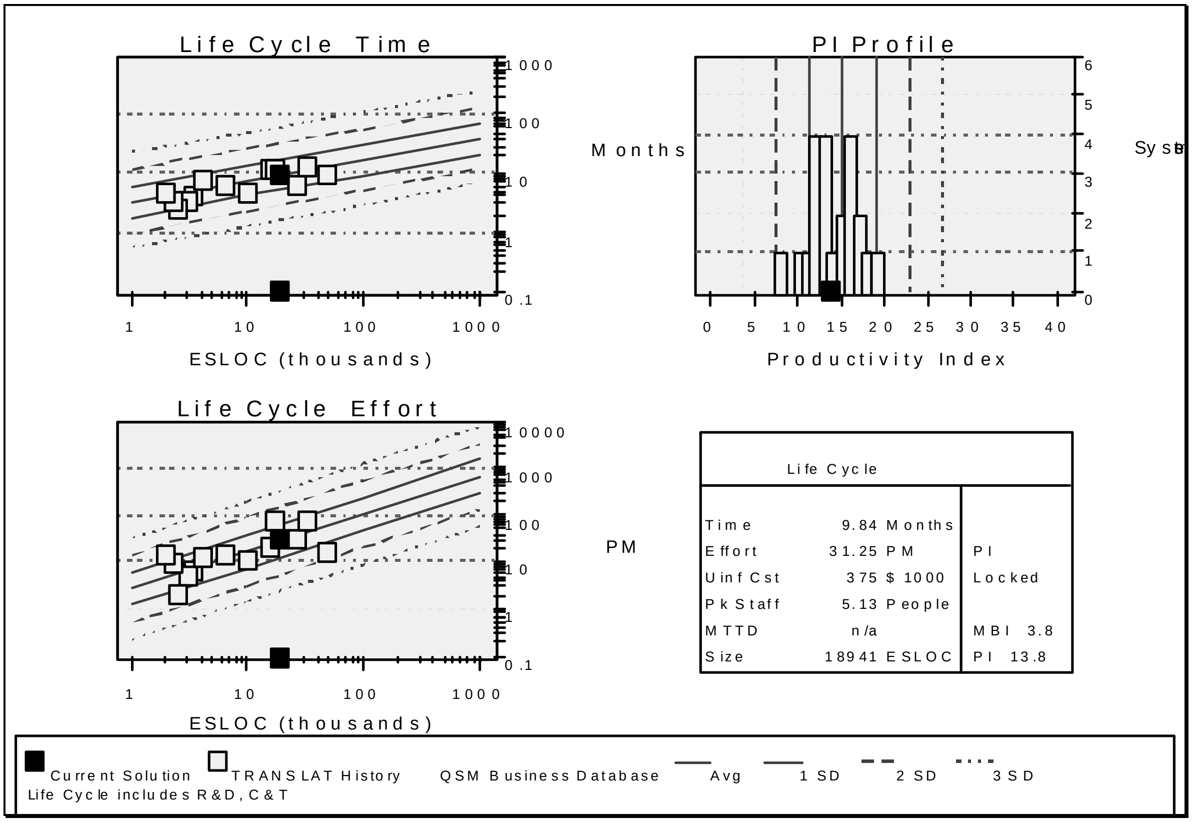


Figure I-24. Translation Estimate Compared to Historic Data in the Ballpark View in SLIM

Notice that the estimate for the Translate Case fits well within the historic data distribution on the schedule and effort graphs. This indicates that the historic data confirms that the plan is achievable. This is further reinforced by looking at the PI profile graph. The PI for our plan is in the middle of the PI distribution of the historic projects again indicating that the plan is consistent with historic performance. If this were a real case the data on these graphs would come from the development organization. The support costs were calculated using the same method as the Redevelopment Case.

Redocumentation

SLIM has a documentation function that estimates the size of documentation products in and the associated effort and cost. The documentation function in SLIM is tunable to historic data. To account for increased documentation in a DoD 2167A environment the tuning parameter was set to 200 %. Figure I-25 shows the breakdown.

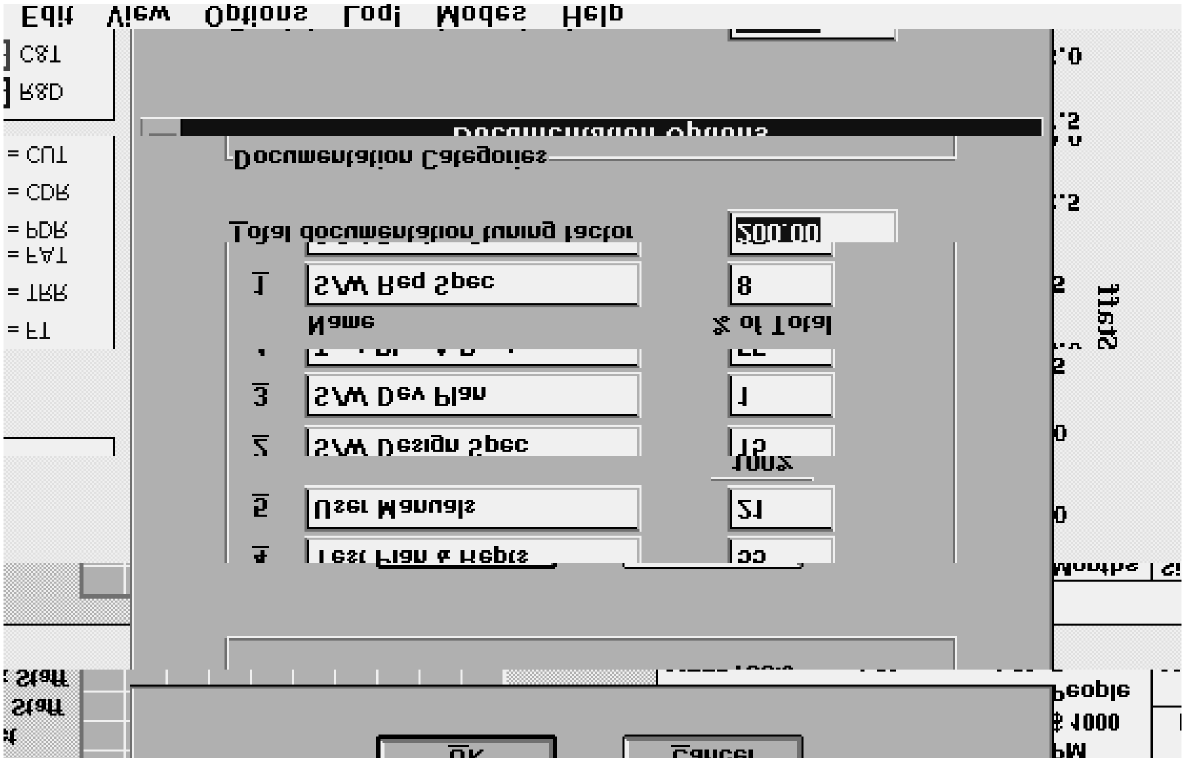


Figure I-25. Documentation Size

SLIM will break out the overall effort by activity. The documentation effort was used in the economic analysis.

I.8 Economic Analysis

In the analysis shown in Figure I-26 note that the Benefit to Investment Ratios (BIR) for the different strategies are very similar. Even though redocumenting the system would produce the highest BIR this might not be the best strategy. Redocumenting the system produces the least Total Benefit and NPV, but also requires the least investment hence the ratio remains essentially unchanged. The BIRs for translating, restructuring and redeveloping the system are statistically the same. The NPV is highest for redevelopment, but statistically close to restructure, Other factors should obviously be considered when making these decisions.

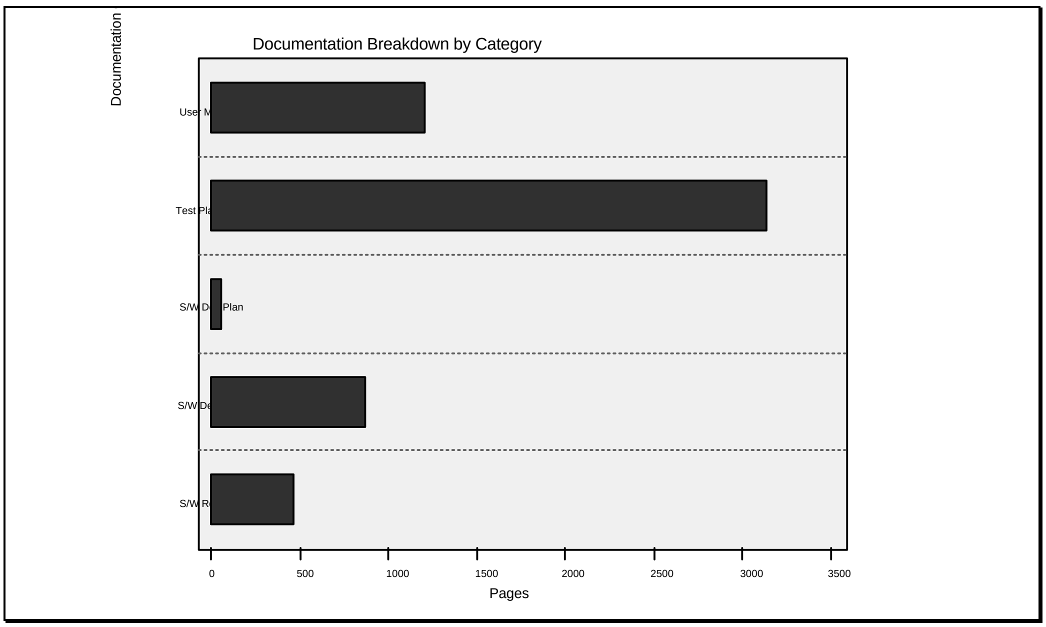


Figure I-26. Economic Analysis

APPENDIX J
SOFTCOST-OO SOFTWARE ESTIMATING MODEL

This appendix describes how the SOFTCOST-OO software estimating model can be applied to reengineering efforts. It was prepared by Mr. Tony Collins of Resource Calculations and edited by Mr. John Clark of Comptek Federal Systems.

Mr. Tony Collins
Resource Calculations, Inc.
7853 East Arapahoe Rd.
Englewood, CO 80112-1361
(303) 267-0379 (Voice)
(303) 694-4869 (Fax)

or

Mr. Tony Collins
Resource Calculations, UK
Rotherglen House
Gerrards Cross Road
Stoke Poges
Bucks SL24EJ
England
011 441 753 662 572 (Voice)
011 441 753 663 398 (Fax)
100135.1155@compuserve.com

J.1 Introduction

SOFTCOST-OO is an extension of, and includes all the features of SOFTCOST-Ada. The Ada estimating features have been replicated for C++ and Object-Oriented estimating. The model is fully described in the SOFTCOST-Ada and SOFTCOST-OO manuals. It utilizes the sophisticated mathematics developed by Robert C. Tausworthe at JPL in 1981 (Publication JPL 81-7). It contains five sub-models:

- Sizing
- Estimation
- Allocation
- Risk
- Life cycle.

For reengineering, the starting point is the development estimate and the complex relationships between the cost drivers. These result in technical adjustment factors which are derived from Architectural and Specification complexity which are major factors in determining life cycle costs and hence the break points in assessing the options to take.

The estimating process is based around the use of Work Breakdown Structure (WBS) formats such as *DOD 2167A*, PRINCE, which is a European based WBS related to LBMS and SSADM structures, or others which can be described in similar form. The model allows for construction of new or modified WBS formats using a special section. When an estimate has been produced, the WBS allocations can be translated into actual Effort and Duration figures to produce a Work Breakdown Actuals (WBA) file. The result is that individual CSUs can be seen in actual days of duration or man-days of effort for each item. This information can then be viewed in a spreadsheet format for each item.

Figures J-1 and J-2 show the WBS and WBA files in their easily readable spreadsheet formats.

The WBA file can also be exported in a variety of formats directly into project planning tools such as MS Project, etc. The parts of the WBS processes which are included in the estimate are shown in Figure J-1. The items included in the estimate are marked with an * in the fourth column. These figures must add up to 100%, and a check routine is included in the WBS section to ensure that when changes are made, the new values retain this integrity. All other items can be assigned a percentage figure over and above the 100%, which, once the estimate has gone firm, will also result in the assignment of effort and duration values. These values are entirely under the command of the user, as is all the data used. The area in the box in Figure J-2 shows what is covered by the estimate. The units in which the effort and duration are expressed are days and man-days.

Once the WBS and Language options have been chosen, sizing can be established. Figure J-3 shows the form of entry for various classes of software. Each line has a help box below the entry area. The one shown is for new software.

There are values in the calibration file for the 'worth' of 'other' lines compared with new, e.g., a line of code which is to be modified may be the equivalent of 0.3 lines of new code in terms of effort and time to process. These values are shown in Figure J-4. They have been derived from a database containing over 300 completed projects.

Up to 99 estimates (strategies) can be set up for any project. This means that for a constant set of cost drivers values, size and distribution of the new or reused code can be investigated. Alternatively the values of the cost drivers other than size can be investigated in a factor sensitivity study. Figure J-5 shows the form of the screen. The help window shows that currently the number of projects (set at 3 in the diagram) is being examined. By changing this to, say, one, the estimate line above will instantaneously change to show the new calculated value for Duration and Effort.

It should be remembered that there is no 'right' estimate - there are an infinite number of valid estimates. Each estimate will, however, have associated with it a confidence factor. Figure J-6 shows how Effort and Duration can be traded against confidence level. When investigating each strategy, it is important to remember that it does not matter what the answer is that is chosen as long as the same confidence level is specified for all the competing strategies.

Life Cycle costs are also covered in the SOFTCOST-OO model. These are based around the original Development Estimate. Figure J-7 shows how the Life Cycle Costs are derived from the original estimates using calculated Technology Adjustment Factors (derived as described earlier from the Cost Drivers of the Development model). Figure J-8 shows the calculation and Figure J-9 how the annualized figures are presented. The screen area in the box can be scrolled sideways to view the full ten year range (used in the example data in Figure J-7.). Inherent in these calculations is a Discounted Cash Flow presentation where the cost of money can be preset.

J.2 Model Inputs

The required input to the SOFTCOST-OO model is a range of factors (cost drivers) as shown in Table J-1. These are divided into 4 sections concerned with Projects, Processes, Product and Personnel. The values entered for these are usually numbers between 1 and 6, the meanings of which are described in the help box below each entry screen for each different entry.

Project Factors • Type of Software • System Architecture • Number of Organizations involved • Organizational Interface Complexity • Staff Resource Availability • Security Requirements	Product Factors • Ada Usage Factor • Requirements Volatility • Degree of Optimization • Degree of Real-Time • Reuse Benefits • Reuse Cost Database Size	The Output Predicts • Effort • Duration • Confidence Level and allows • Sensitivity Analysis for all drivers individually • Examination of ranges of confidence level for risk analysis
Process Factors • Degree of Standardization • Scope of Support • Use of Modern Software Methods • Use of Peer Reviews • Use of Software Tools/ Environment • Software Tool/Environment Stability	Personnel Factors • Number of Ada Projects Completed by the team • Analyst Capability • Applications Experience • Ada Environment Experience • Ada Language Experience • Ada Methodology Experience • Team Capability Sizing and Sub-Projects • KSLOCS or Function Points • Up to 9 Sub-Projects with Inherited or Unique characteristics	• Printing of GANTT & PERT charts for Work Breakdown Structures which can use a standard default (*DoD 2167*) or user defined Access to Data • All Drive Parameters, WBS Structures, Holiday Tables, etc. can be redefined by the user • Export to Project Life Cycle (LC) model and to external Project Management packages

Table J-1. SOFTCOST-OO Factors (Cost Drivers)

J.3 Baseline Estimates - Status Quo

SOFTCOST-OO contains a Life Cycle Model which, based on a development Effort and Duration plus assumptions on: (a) the level of Sustaining Engineering Staff (they deal with that part of maintenance which is concerned with keeping the unchanged application running, e.g., where environment changes, user support, training, and unscheduled repairs have to be covered), (b) on the level of maintenance staff (those covering changes and enhancement of the system), (c) on Annual Change Traffic (increase in functionality or change of requirements), and (d) on product life and cost of money, gives the net present value of the on-going costs. These on-going costs are described in this handbook's parlance as Operations and Support (O&S) respectively.

J.4 Redocumentation

SOFTCOST-OO works around the items in a Work Breakdown Structure. The WBS will contain the necessary CSI's to describe the process which is being undertaken. Typically *DoD 2167A* contains all the documentation items. SOFTCOST-OO can produce an estimate and reflect this estimate into the WBS file to produce a new WBA file which contains the cost elements for all items. The fractional percentage figures for duration and effort for each item of the WBS are not immutable, provided that the total of the included items remains 100%. The WBS system contains checks to insure: (a) that the figures add up to 100% (by pressing F5) and, (b) that there are no “loops” in the predecessors (by pressing F7). From these the documentation items can be isolated and summed to obtain the estimated redocumentation costs.

J.5 Restructure

The main model has a section where the size of the changes to the system are entered. These can be in either KSLOCs or Function Points (based on Extended Albrecht counting conventions). The subheadings are:

- New Code to be produced
- Reused Code (unaltered but retested)
- Code to be modified (and retested).

These categories are specified for both the main language (Ada, C++ or generic Object Oriented paradigm) and/or for 'other' language components. The model is then run to determine the development cost and timescale, and the Life Cycle model draws directly on the results of that estimate to produce the Life Cycle costs using the given product life and selected cost of money.

J.6 Translation

SOFTCOST-OO works in one of two ways :

- The number of KSLOCs for each language can be entered. This will enable Life Cycle costs for each option to be calculated. For the new language, the development cost to add to this will also be produced.

- It may be necessary to consider changing some of the cost drivers (e.g., to take account of differing experience levels in the development staff, different team practices in new developments compared with what was done in the old, etc.)

J.7 Retargeting

The Cost Drivers in SOFTCOST-OO allow for changes in Hardware configuration. The model would be run for both old and new configurations to estimate the Life Cycle costs for each, and the development cost of the new configuration would also be produced to add to the LC cost to give the comparison.

J.8 Data Reengineering

SOFTCOST-OO is sensitive to the size of the database but not directly to the technology. There would probably be some changes to other cost drivers such as System Architecture, Degree of Standardization or Product Complexity. Whether there were changes of this type or not, the model would probably also need some programming code changes also, so the model would need to be run to determine both development cost and changed life cycle cost.

J.9 Reverse Engineering

Here a new estimate with, probably, a totally new set of Cost Drivers would need to be run to give both development and LC costs for comparison with on-going costs.

J.10 Reverse Engineering With Other Strategies

Here, as many runs of the model as necessary could be undertaken. It would be important to run the Project Summary Report, which details all the assumptions made and records all outputs for each estimate, so that the comparisons could be made once all option results had been obtained.

J.11 Redevelopment

Once again, all this would require would be a rerun of SOFTCOST-OO with the new parameter set determined by the actual engineering process chosen. Both Development and Life Cycle costs are a natural output from this process.

J.12 Operations & Support

As explained in Section J.3 the Operations and Support elements naturally fall out of the Life Cycle model. Before and after runs are, therefore, all that are necessary to produce all the data required.

Task Code	Title	Parent	Effort	Duration	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5
1.	SYS REQT ANALYS/DSGN PHASE		0.00	0					
1.1	SW DEVELOPMENT TASKS 1	1.	0.00	0	1.				
1.1.1	SUPPORT SS DEV	1.1	0.50	2.22	1.1				
1.1.2	SUPPORT OCD DEV	1.1	0.50	2.22					
1.1.3	SUPPORT SRR	1.1	0.50	0.56	1.1.1	1.1.2			
1.1.4	SUPPORT SSDD DEV	1.1	2.00	1.67	1.1.3				
1.1.5	DEVELOP PREL SRS(S)	1.1	3.50	1.67	1.1.3				
1.1.6	DEVELOP PREL IRS(S)	1.1	2.00	1.67	1.1.3				
1.1.7	SUPPORT STP DEV	1.1	0.50	1.67	1.1.3				
1.1.8	SUPPORT SDR	1.1	0.50	1.11	1.1.4	1.1.5	1.1.6	1.1.7	
1.1.9	SDR COMPLETED	1.1	0.00	0	1.1.8				
1.2	SW MANAGEMENT TASKS 1	1.	0.00	0	1.				
1.2.1	PREPARE SDP	1.2	1.00	5.56	1.2				
1.3	SCM TASKS 1	1.	0.00	0	1.				
1.3.1	PREPARE CM PLAN	1.3	0.50	5.56	1.3				
1.4	SQE TASKS 1	1.	0.00	0	1.				
1.4.1	PREPARE QE PLAN	1.4	0.50	5.56	1.4				
2.	SW REQTS ANALYSIS PHASE		0.00	0	1.1.9	1.2.1	1.3.1	1.4.1	
2.1	SW DEVELOPMENT TASKS 2	2.	0.00	0	2.				
2.1.1	DEVELOP SRS(S)	2.1	*5	5.00	2.1				
2.1.2	DEVELOP IRS(S)	2.1	*3	5.00	2.1				
2.1.3	DEVELOP PREL MANUALS	2.1	*2.0	3.53	2.1				
2.1.4	ACQUIRE/READY SEE	2.1	*5.0	5.00	2.1				
2.1.5	CONDUCT TRAINING	2.1	*2.0	2.94	2.1				
2.1.6	CONDUCT SSR	2.1	*3.0	1.47	2.1.5				
2.1.7	SSR COMPLETED	2.1	0.00	0	2.1.1	2.1.2	2.1.3	2.1.4	2.1.6

Task Code	Title	Parent	Effort	Duration	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5
2.2	SW MANAGEMENT TASKS 2	2.	0.00	0	2.				
2.3	SCM TASKS 2	2.	0.00	0	2.				
2.3.1	DEVELOP CM PROC	2.3	0.25	0.93	2.3				
2.3.2	OPERATE SW LIBRARY 2	2.3	0.50	2.80	2.3.1				
2.4	SQE TASKS 2	2.	0.00	0	2.				
2.4.1	DEVELOP QE PROC	2.4	0.50	0.93	2.4				
2.4.2	CONDUCT EVALUATIONS 2	2.4	0.50	2.80	2.4.1				
3.	SW PREL DESIGN PHASE		0.00	0	2.1.7	2.2.1	2.3.2	2.4.2	
3.1	SW DEVELOPMENT TASKS 3	3.	0.00	0	3.				
3.1.1	DEVELOP SDD(S)	3.1	*5.0	3.53	3.1				
3.1.2	DEVELOP IDD(S)	3.1	*3.0	3.53	3.1				
3.1.3	DEVELOP STP	3.1	*4.0	3.53	3.1				
3.1.4	CONDUCT PDR	3.1	*3.0	1.47	3.1				
3.1.5	PDR COMPLETED	3.1	0.00	0	3.1.1	3.1.2	3.1.3	3.1.4	
3.2	SW MANAGEMENT TASKS 3	3.	0.00	0	3.				
3.2.1	CONDUCT REVIEWS 3	3.2	1.50	2.05P2	3.2				
3.3	SCM TASKS 3	3.	0.00	0	3.				
3.3.1	OPERATE SW LIBRARY 3	3.3	0.75	2.80	3.3				
3.4	SQE TASKS 3	3.	0.00	0	3.				
3.4.1	CONDUCT EVALUATIONS 3	3.4	1.50	2.80	3.4				
4.	SW DETAILED DESIGN PHASE		0.00	0	3.1.5	3.2.1	3.3.1	3.4.1	
4.1	SW DEVELOPMENT TASKS 4	4.	0.00	0	4.				
4.1.1	DETAIL SDD(S)	4.1	*5.0	3.53	4.1				
4.1.2	DETAIL IDD(S)	4.1	*2.0	3.53	4.1				
4.1.3	DEVELOP TEST CASE(S)	4.1	*3.0	3.53	4.1				

Task Code	Title	Parent	Effort	Duration	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5
4.1.4	CONDUCT PEER REVIEWS 4	4.1	*2.0	2.94	4.1				
4.1.5	CONDUCT CDR	4.1	*3.0	1.47	4.1.4				
4.1.6	CDR COMPLETED	4.1	0.00	0	4.1.1	4.1.2	4.1.3	4.1.5	
4.2	SW MANAGEMENT TASKS 4	4.	0.00	0	4.				
4.2.1	CONDUCT REVIEWS 4	4.2	1.50	2.11P2	4.2				
4.3	SCM TASKS 4	4.	0.00	0	4.				
4.3.1	OPERATE SW LIBRARY 4	4.3	1.00	2.87	4.3				
4.4	SQE TASKS 4	4.	0.00	0	4.				
4.4.1	CONDUCT EVALUATIONS 4	4.4	1.50	2.87	4.4				
5.	CODING & CSU TESTING PHASE		0.00	0	4.1.6	4.2.1	4.3.1	4.4.1	
5.1	SW DEVELOPMENT TASKS 5	5.	0.00	0	5.				
5.1.1	CODE & TEST UNITS	5.1	*10.0	5.61	5.1				
5.1.2	CONDUCT PEER REVIEWS 5	5.1	*3.0	5.61	5.1				
5.1.3	CONDUCT UTR	5.1	*2.0	2.34	5.1				
5.1.4	UTR COMPLETED	5.1	0.00	0	5.1.1	5.1.2	5.1.3		
5.2	SW MANAGEMENT TASKS 5	5.	0.00	0	5.				
5.2.1	CONDUCT REVIEWS 5	5.2	1.50	2.11P2	5.2				
5.3	SCM TASKS 5	5.	0.00	0	5.				
5.3.1	OPERATE SW LIBRARY 5	5.3	1.00	2.87	5.3				
5.4	SQE TASKS 5	5.	0.00	0	5.				
5.4.1	CONDUCT EVALUATIONS 5	5.4	1.50	2.87	5.4				
6.	CSC INT & TESTING PHASE		0.00	0	5.1.4	5.2.1	5.3.1	5.4.1	
6.1	SW DEVELOPMENT TASKS 6	6.	0.00	0	6.				
6.1.1	INT & TEST SW	6.1	*17.0	10.28	6.1				
6.1.2	DEVELOP STD(S)	6.1	*5.0	9.35	6.1				
6.1.3	CONDUCT TRR	6.1	*3.0	2.34	6.1.1	6.1.2			

Task Code	Title	Parent	Effort	Duration	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5
6.1.4	TRR COMPLETED	6.1	0.00	0	6.1.1	6.1.3			
6.2	SW MANAGEMENT TASKS 6	6.	0.00	0	6.				
6.2.1	CONDUCT REVIEWS 6	6.2	2.50	4.22P1	6.2				
6.3	SCM TASKS 6	6.	0.00	0	6.				
6.3.1	OPERATE SW LIBRARY 6	6.3	1.50	4.79	6.3				
6.4	SQE TASKS 6	6.	0.00	0	6.				
6.4.1	CONDUCT EVALUATIONS 6	6.4	2.00	4.79	6.4				
7.	CSCI TESTING PHASE		0.00	0	6.1.4	6.2.1	6.3.1	6.4.1	
7.1	SW DEVELOPMENT TASKS 7	7.	0.00	0	7.				
7.1.1	CONDUCT SW T & E	7.1	*3.0	3.27	7.1				
7.1.2	DEVELOP STR(S)	7.1	*2.0	3.27	7.1				
7.1.3	DEVELOP SPS(S)	7.1	*2.0	3.27	7.1				
7.1.4	DEVELOP O&S DOCUMENTS	7.1	*2.0	3.27	7.1				
7.1.5	CONDUCT SW FCA/PCA	7.1	*1.0	1.39	7.1.1	7.1.2	7.1.3	7.1.4	
7.1.6	SW FCA/PCA COMPLETED	7.1	0.00	0	7.1.5				
7.2	SW MANAGEMENT TASKS 7	7.	0.00	0	7.				
7.2.1	CONDUCT REVIEWS 7	7.2	1.00	4.44P1	7.2				
7.3	SCM TASKS 7	7.	0.00	0	7.				
7.3.1	OPERATE SW LIBRARY 7	7.3	1.00	5.56	7.3				
7.4	SQE TASKS 7	7.	0.00	0	7.				
7.4.1	CONDUCT EVALUATIONS 7	7.4	1.00	5.56	7.4				
8.	SYSTEM INT & TEST PHASE		0.00	0	7.1.6	7.2.1	7.3.1	7.4.1	
8.1	SW DEVELOPMENT TASKS 8	8.	0.00	0	8.				
8.1.1	SUPPORT SYSTEM T&E	8.1	25.00	11.11	8.1				
8.1.2	PREPARE PRODUCT BASELINE	8.1	4.00	2.78	8.1.1				
8.1.3	SUPPORT SYSTEM FCA/PCA	8.1	1.00	2.78	8.1.1				

Task Code	Title	Parent	Effort	Duration	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5
8.1.4	SYSTEM FCA/PCA COMPLETED	8.1	0.00	0	8.1.2	8.1.3			
8.2	SW MANAGEMENT TASKS 8	8.	0.00	0	8.				
8.2.1	CONDUCT REVIEWS 8	8.2	3.00	10.56P2	8.2				
8.3	SCM TASKS 8	8.	0.00	0	8.				
8.3.1	OPERATE SW LIBRARY 8	8.3	0.50	13.89	8.3				
8.4	SQE TASKS 8	8.	0.00	0	8.				
8.4.1	CONDUCT EVALUATIONS 8	8.4	1.00	13.89	8.4	8.2.1	8.3.1		

Figure J-2. Work Breakdown Actual (WBA)

Task Code	Title	Parent	Effort	Duration	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5
1.	SYS REQT ANALYS/DSGN PHASE		0.00						
1.1	SW DEVELOPMENT TASKS 1	1.	0.00		1.				
1.1.1	SUPPORT SS DEV	1.1	25.20		1.1				
1.1.2	SUPPORT OCD DEV	1.1	25.20						
1.1.3	SUPPORT SRR	1.1	6.36		1.1.1	1.1.2			
1.1.4	SUPPORT SSDD DEV	1.1	18.95		1.1.3				
1.1.5	DEVELOP PREL SRS(S)	1.1	18.95		1.1.3				
1.1.6	DEVELOP PREL IRS(S)	1.1	18.95		1.1.3				
1.1.7	SUPPORT STP DEV	1.1	18.95		1.1.3				
1.1.8	SUPPORT SDR	1.1	12.60		1.1.4	1.1.5	1.1.6	1.1.7	
1.1.9	SDR COMPLETED	1.1	0.00		1.1.8				
1.2	SW MANAGEMENT TASKS 1	1.	0.00		1.				
1.2.1	PREPARE SDP	1.2	63.11		1.2				
1.3	SCM TASKS 1	1.	0.00		1.				
1.3.1	PREPARE CM PLAN	1.3	63.11		1.3				
1.4	SQE TASKS 1	1.	0.00		1.				
1.4.1	PREPARE QE PLAN	1.4	63.11		1.4				
2.	SW REQTS ANALYSIS PHASE		0.00		1.1.9	1.2.1	1.3.1	1.4.1	
2.1	SW DEVELOPMENT TASKS 2	2.	0.00		2.				
2.1.1	DEVELOP SRS(S)	2.1	*		2.1				
2.1.2	DEVELOP IRS(S)	2.1	*		2.1				
2.1.3	DEVELOP PREL MANUALS	2.1	*		2.1				
2.1.4	ACQUIRE/READY SEE	2.1	*		2.1				
2.1.5	CONDUCT TRAINING	2.1	*		2.1				
2.1.6	CONDUCT SSR	2.1	*		2.1.5				
2.1.7	SSR COMPLETED	2.1	0.00		2.1.1	2.1.2	2.1.3	2.1.4	2.1.6

Task Code	Title	Parent	Effort	Duration	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5
2.2	SW MANAGEMENT TASKS 2	2.	0.00		2.				
2.2.1	CONDUCT REVIEWS 2	2.2	*		2.2				
2.3	SCM TASKS 2	2.	0.00		2.				
2.3.1	DEVELOP CM PROC	2.3	*		2.3				
2.3.2	OPERATE SW LIBRARY 2	2.3	*		2.3.1				
2.4	SQE TASKS 2	2.	0.00		2.				
2.4.1	DEVELOP QE PROC	2.4	*		2.4				
2.4.2	CONDUCT EVALUATIONS 2	2.4	*		2.4.1				
3.	SW PREL DESIGN PHASE		0.00		2.1.7	2.2.1	2.3.2	2.4.2	
3.1	SW DEVELOPMENT TASKS 3	3.	0.00		3.				
3.1.1	DEVELOP SDD(S)	3.1	*		3.1				
3.1.2	DEVELOP IDD(S)	3.1	*		3.1				
3.1.3	DEVELOP STP	3.1	*		3.1				
3.1.4	CONDUCT PDR	3.1	*		3.1				
3.1.5	PDR COMPLETED	3.1	0.00		3.1.1	3.1.2	3.1.3	3.1.4	
3.2	SW MANAGEMENT TASKS 3	3.	0.00		3.				
3.2.1	CONDUCT REVIEWS 3	3.2	*		3.2				
3.3	SCM TASKS 3	3.	0.00		3.				
3.3.1	OPERATE SW LIBRARY 3	3.3	*		3.3				
3.4	SQE TASKS 3	3.	0.00		3.				
3.4.1	CONDUCT EVALUATIONS 3	3.4	*		3.4				
4.	SW DETAILED DESIGN PHASE		0.00		3.1.5	3.2.1	3.3.1	3.4.1	
4.1	SW DEVELOPMENT TASKS 4	4.	0.00		4.				
4.1.1	DETAIL SDD(S)	4.1	*		4.1				
4.1.2	DETAIL IDD(S)	4.1	*		4.1				
4.1.3	DEVELOP TEST CASE(S)	4.1	*		4.1				

Task Code	Title	Parent	Effort	Duration	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5
4.1.4	CONDUCT PEER REVIEWS 4	4.1	*		4.1				
4.1.5	CONDUCT CDR	4.1	*		4.1.4				
4.1.6	CDR COMPLETED	4.1	0.00		4.1.1	4.1.2	4.1.3	4.1.5	
4.2	SW MANAGEMENT TASKS 4	4.	0.00		4.				
4.2.1	CONDUCT REVIEWS 4	4.2	*		4.2				
4.3	SCM TASKS 4	4.	0.00		4.				
4.3.1	OPERATE SW LIBRARY 4	4.3	*		4.3				
4.4	SQE TASKS 4	4.	0.00		4.				
4.4.1	CONDUCT EVALUATIONS 4	4.4	*		4.4				
5.	CODING & CSU TESTING PHASE		0.00		4.1.6	4.2.1	4.3.1	4.4.1	
5.1	SW DEVELOPMENT TASKS 5	5.	0.00		5.				
5.1.1	CODE & TEST UNITS	5.1	*		5.1				
5.1.2	CONDUCT PEER REVIEWS 5	5.1	*		5.1				
5.1.3	CONDUCT UTR	5.1	*		5.1				
5.1.4	UTR COMPLETED	5.1	0.00		5.1.1	5.1.2	5.1.3		
5.2	SW MANAGEMENT TASKS 5	5.	0.00		5.				
5.2.1	CONDUCT REVIEWS 5	5.2	*		5.2				
5.3	SCM TASKS 5	5.	0.00		5.				
5.3.1	OPERATE SW LIBRARY 5	5.3	*		5.3				
5.4	SQE TASKS 5	5.	0.00		5.				
5.4.1	CONDUCT EVALUATIONS 5	5.4	*		5.4				
6.	CSC INT & TESTING PHASE		0.00		5.1.4	5.2.1	5.3.1	5.4.1	
6.1	SW DEVELOPMENT TASKS 6	6.	0.00		6.				
6.1.1	INT & TEST SW	6.1	*		6.1				
6.1.2	DEVELOP STD(S)	6.1	*		6.1				
6.1.3	CONDUCT TRR	6.1	*		6.1.1	6.1.2			

Task Code	Title	Parent	Effort	Duration	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5
6.1.4	TRR COMPLETED	6.1	0.00		6.1.1	6.1.3			
6.2	SW MANAGEMENT TASKS 6	6.	0.00		6.				
6.2.1	CONDUCT REVIEWS 6	6.2	*		6.2				
6.3	SCM TASKS 6	6.	0.00		6.				
6.3.1	OPERATE SW LIBRARY 6	6.3	*		6.3				
6.4	SQE TASKS 6	6.	0.00		6.				
6.4.1	CONDUCT EVALUATIONS 6	6.4	*		6.4				
7.	CSCI TESTING PHASE		0.00		6.1.4	6.2.1	6.3.1	6.4.1	
7.1	SW DEVELOPMENT TASKS 7	7.	0.00		7.				
7.1.1	CONDUCT SW T & E	7.1	*		7.1				
7.1.2	DEVELOP STR(S)	7.1	*		7.1				
7.1.3	DEVELOP SPS(S)	7.1	*		7.1				
7.1.4	DEVELOP O&S DOCUMENTS	7.1	*		7.1				
7.1.5	CONDUCT SW FCA/PCA	7.1	*		7.1.1	7.1.2	7.1.3	7.1.4	
7.1.6	SW FCA/PCA COMPLETED	7.1	0.00		7.1.5				
7.2	SW MANAGEMENT TASKS 7	7.	0.00		7.				
7.2.1	CONDUCT REVIEWS 7	7.2	*		7.2				
7.3	SCM TASKS 7	7.	0.00		7.				
7.3.1	OPERATE SW LIBRARY 7	7.3	*		7.3				
7.4	SQE TASKS 7	7.	0.00		7.				
7.4.1	CONDUCT EVALUATIONS 7	7.4	*		7.4				
8.	SYSTEM INT & TEST PHASE		0.00		7.1.6	7.2.1	7.3.1	7.4.1	
8.1	SW DEVELOPMENT TASKS 8	8.	0.00		8.				
8.1.1	SUPPORT SYSTEM T&E	8.1	126.10		8.1				
8.1.2	PREPARE PRODUCT BASELINE	8.1	31.55		8.1.1				
8.1.3	SUPPORT SYSTEM FCA/PCA	8.1	31.55		8.1.1				

Task Code	Title	Parent	Effort	Duration	Pred 1	Pred 2	Pred 3	Pred 4	Pred 5
8.1.4	SYSTEM FCA/PCA COMPLETED	8.1	0.00		8.1.2	8.1.3			
8.2	SW MANAGEMENT TASKS 8	8.	0.00		8.				
8.2.1	CONDUCT REVIEWS 8	8.2	119.86		8.2				
8.3	SCM TASKS 8	8.	0.00		8.				
8.3.1	OPERATE SW LIBRARY 8	8.3	157.65		8.3				
8.4	SQE TASKS 8	8.	0.00		8.				
8.4.1	CONDUCT EVALUATIONS 8	8.4	157.65		8.4	8.2.1	8.3.1		

F1 PROJECT FACTORS	F2 PROCESS FACTORS	F3 PRODUCT FACTORS	F4 PERSONNEL FACTORS	F5 SUBPROJECT SETUP	F6 SIZING METHOD	F7 PROJECT REPORTS	F10 EXIT

DEMO(1) SOFTCOST-OO SIZING FACTORS
KILO-LINES OF SOURCE CODE WITH A 99% CONFIDENCE LEVEL

How many lines of source code will be:			MAX	MOST LIKELY	MIN
Components	1)	new software?	50.00	50.00	50.00
	2)	re-used software?	10.00	10.00	10.00
	3)	modified software?	5.00	5.00	5.00
Other Components:	1)	new software?	0.00	0.00	0.00
	2)	re-used software?	0.00	0.00	0.00
	3)	modified software?	0.00	0.00	0.00

Object Oriented Language Components (New):
Enter the number of thousand source lines of new OO code (exclude) comments, but include data declarations, instantiations and program specifications). An OO source line is defined by a terminal semicolon with instantiated code counted once.

Figure J-3. SOFTCOST-OO Sizing Factors

F1 ESTIMATE REPORT	F2 FACTOR SENSITIVITY	F3 WHAT-IF GAMING	F4 RESOURCE ALLOCATION	F5 PROJECT SUMMARY	F6 GANTT & PERT CHARTS	F7 PRINT	F10 EXIT

SOFTCOST-OO PROJECT SUMMARY REPORT

SIZING SUB-MODEL WEIGHTING FACTORS

New OO Components	: 1.000
Re-used OO Components	: 0.200
Modified OO Components	: 0.300
New Other Components	: 1.000
Re-used Other Components	: 0.250
Modified Other Components	: 0.400

	Effective Size (KSLOC/FP)	Duration (months)	Effort (pm)	Productivity (SLOC/pm)	Average Staff (persons)	Confidence (%)
DEMO	124.7	24.7	492.0	225.1	19.9	39.9
DEMO(1)	52.5	20.1	229.8	208.6	11.4	42.2

Press ↑ ↓ to scroll. Press a function key to select next report.

Figure J-4. SOFTCOST-OO Project Summary Report

F1 PROJECT FACTORS	F2 PROCESS FACTORS	F3 PRODUCT FACTORS	F4 PERSONNEL FACTORS	F5 SUBPROJECT SETUP	F6 SIZING FACTORS	F7 PROJECT REPORTS	F10 EXIT

Project Estimate:	Size	Duration	Effort	Productivity
	124.7 KSLOC	24.7 Months	492.0 PM	225.1 SLOC/PM

DEMO SOFTCOST-OO PERSONNEL FACTORS

1)	Number of OO Projects Completed by Team:	3
2)	Analyst Capability:	3
3)	Applications Experience:	4
4)	Environment Experience:	3
5)	Language Experience:	3
6)	Methodology Experience:	3
7)	Team Capability:	4

Experience Profile:
Enter the average number of OO projects completed by the team that will be assigned to this project. A project is defined as a delivery of a product packaged and prepared using object-oriented concepts.

Figure J-5. SOFTCOST-OO Personnel Factors

F1 ESTIMATE REPORT	F2 FACTOR SENSITIVITY	F3 WHAT-IF GAMING	F4 RESOURCE ALLOCATION	F5 PROJECT SUMMARY	F6 GANTT & PERT CHARTS	F7 PRINT	F10 EXIT

RESOURCE ALLOCATION SUMMARY (LABOR HOURS)
DEMO WBS File: STANDARD.WBS

Life Cycle Phases	SW Development Effort	SW Management Effort	SCM Effort	SQE Effort
SYS REQT ANALYS/DSGN PHASE	7872.0	787.2	393.6	393.6
SW REQTS ANALYSIS PHASE	15744.0	1574.4	590.4	787.2
SW PREL DESIGN PHASE	11808.0	1180.0	590.4	1180.8
SW DETAILED DESIGN PHASE	11808.0	1180.0	787.2	1180.8
CODING & CSU TESTING PHASE	11808.0	1180.0	787.2	1180.8
CSC INT & TESTING PHASE	19680.0	1968.0	1180.8	1574.4
CSCI TESTING PHASE	7872.0	787.2	787.2	787.2
SYSTEM INT & TEST PHASE	23616.0	2361.6	393.6	787.2
TOTALS	110208.0	11020.8	5510.4	7872.0

Press any key to display Resource Allocation Graph

Figure J-6. Resource Allocation Summary

F1 ESTIMATE REPORT	F2 FACTOR SENSITIVITY	F3 WHAT-IF GAMING	F4 RESOURCE ALLOCATION	F5 PROJECT SUMMARY	F6 GANTT & PERT CHARTS	F7 PRINT	F10 EXIT

DEMO SOFTCOST-OO "WHAT-IF" PLOT CALCULATIONS

EFFORT	DURATION	STAFF	PRODUCTIVITY	CONFIDENCE
492.0	24.7	19.9	225.1	39.9
565.8	24.7	22.9	221.3	49.9
639.6	24.7	25.9	218.0	57.9
713.2	24.7	28.9	215.1	64.2
787.4	24.7	31.9	212.6	69.3
861.0	24.7	34.9	210.3	73.4
934.8	24.7	37.8	208.2	76.7
1008.6	24.7	40.8	206.2	79.4
1082.4	24.7	43.8	204.5	81.7
1156.2	24.7	46.8	202.8	83.7
1230.0	24.7	49.8	201.3	85.3

Press any key to display "What-If" Plot

Figure J-7. SOFTCOST-OO "What-If" Plot Calculations

F1 ADA-MODEL VALUES	USER-SPECIFIED VALUES		F7 LIFE-CYCLE REPORTS	F10 EXIT

DEMO SOFTCOST-OO LIFE-CYCLE FACTORS

1) Nominal Effort (person-months):	492.0
2) Nominal Duration (months):	24.7
3) Sustaining engineering factor (%):	5.0
4) Annual change traffic (%):	10.0
5) Product Life (years):	10
6) Cost of money (%):	5.0

Press <ENTER> to continue or any other key to edit:

Figure J-8. SOFTCOST-OO Life-Cycle Factors

F1 ESTIMATE REPORT	F2 RESOURCE LOADING	F3 LOAD BALANCING	F4 PROJECT SUMMARY	F5 RATE OPTIONS	F6 LIFE-CYCLE FACTORS	F7 PRINT	F10 EXIT

DEMO SOFTCOST-OO LIFE-CYCLE ESTIMATE

Base Effort (PM)	Effort Adjustment Factor	Sustaining Engineering Factor	Annual Change Traffic	Average O&S Phase Effort (PM/YR)
492.0	0.8	5%	10%	94.3

Life of product (years): 10
Calibration coefficient: 1.4

Press a function key to select next report.

Figure J-9. SOFTCOST-OO Life-Cycle Estimate

F1 ESTIMATE REPORT	F2 RESOURCE LOADING	F3 LOAD BALANCING	F4 PROJECT SUMMARY	F5 RATE OPTIONS	F6 LIFE-CYCLE FACTORS	F7 PRINT	F10 EXIT

DEMO RESOURCE LOADING SUMMARY

	Year 1	Year 2	Year 3	Year 4	Year 5
Total Staff Level	9.4	8.6	7.9	7.9	7.9
Maintenance Staff	5.9	5.4	4.9	4.9	4.9
Sustaining Staff	3.5	3.2	2.9	2.9	2.9
Annual Cost (X 1,000)	905.5	830.1	754.6	754.6	754.6
Cost of Money (%)	5.0	5.0	5.0	5.0	5.0
Present Value (X 1,000)	862.4	752.9	651.9	620.8	591.2

Use → to scroll. Press a function key to continue.

Figure J-10. Resource Loading Summary

APPENDIX K

SRAH HISTORY

The Joint Logistics Commanders (JLC) Joint Policy Coordinating Group (JPCG) on Computer Resources Management (CRM) initiated this handbook at the First Software Reengineering Workshop in September 1992 at Santa Barbara, CA (Santa Barbara I). The draft handbook, entitled *Reengineering Economics Handbook (REH)*, was developed by the members of the Reengineering Economics Panel during the workshop, whose co-chairs and members were selected from all services and industry.

Workshop Chairman: Mr. Jim Mohan, Naval Air Warfare Center (NAWC)

Panel Co-Chairs: Mr. John Clark, Comptek
Mr. Chris Sittenauer, USAF/STSC

Panel Members:

Col. John Barnhart, SSC	Capt. Mike McPherson, USAF
Mr. John Donald, AFCAA	Mr. Mike Olsem, STSC/SAIC
Mr. William Head, USMC	Mr. Eric Shulman, USAF
Mr. Lester Hummel, USN	Mr. Barry Stevens, Comptek
Capt. Mark Kanko, USAF	Mr. Tom Vigorito, Comptek
Mr. Ben Keller, VA Tech	Mr. Mike Wood
Mr. John LeBaron, USA	

Following the workshop, a subgroup of the original Reengineering Economics Panel continued working to complete and further refine the handbook. This subgroup consisted of the following personnel:

Col. John Barnhart, SSC	Capt. Mike McPherson, USAF
Mr. John Clark, Comptek	Mr. Mike Olsem, STSC/SAIC
Mr. John Donald, AFCAA	Mr. Chris Sittenauer, USAF/STSC
Capt. Mark Kanko, USAF	Mr. Barry Stevens, Comptek
	Mr. Mike Wood

Mr. John Clark and Mr. Barry Stevens of Comptek Federal Systems were tasked in April 1993 by the Air Force Cost Analysis Agency (AFCAA) to refine the technical assessment process for Version 1.0. In addition, Mr. Henry Apgar and Ms. Sherry Stukes of Management Consulting and Research were tasked by AFCAA to develop the economic assessment process and provide overall integration and editing of Version 1.0. Recognizing the dual contributions of the technical and economic analyses to the overall assessment process, the title of the handbook was changed to Software Reengineering Assessment Handbook (SRAH).

Version 1.0 was published in February 1994 with Mr. John Donald of the Air Force Cost Analysis Agency (AFCAA) serving as the focal point for the project. Table K-1 lists the personnel who provided over 1,200 comments on Version 1.0. The comments received on Version 1.0 were evaluated and nearly all of them were included in Version 2.0. (Schedule and financial limitations prevented implementation of some major elements such as reuse economics and systems reengineering, which are candidates for future versions.) Mr. Mike Olsem (STSC/SAIC) and Mr. Chris Sittenauer (USAF/STSC) completely rewrote the Technical Assessment Process (Section 4) and the Management Decision Process (Section 6) respectively. Mr. Dennis Barney (STSC/TRW) and Mr. Alan Giles (STSC/TRW) collaborated with Mr. John Clark (Comptek) to rewrite the Economic Assessment Process (Section 5). Great appreciation is extended to these personnel for their significant contributions to this handbook.

SRAH Version 3.0 was begun in early 1996 and completed by early 1997. The intent of Version 3.0 was to include all additional comments by SRAH users after the Handbook had been in use for a couple of years. Cost Models were updated and coordinated through the efforts of Mr. John Clark (Comptek). The Management Decision Process (Section 6) was clarified and rewritten by Mr. Mike Olsem (STSC/SAIC). The Executive Summary Section was modified by Mr. Bryce Ragland (USAF/STSC).

Versions 2.0 and 3.0 were produced due to the continued efforts of the subgroup, with Mr. Robert Johnson, (leader of the JGSE Reengineering & Reuse (R&R) Functional Working Group (FWG)), serving as the focal point for the project. Special thanks to Ms. Karen Rasmussen (STSC/SAIC) and Mrs. Lisa Fisher (STSC/SAIC) for their editing and coordination efforts to put SRAH Versions 2.0 and 3.0 into a publishable format. The JGSE R&R FWG includes the following participants, all of whom contributed to Version 2.0:

Ms. Susan Crosswait, MCTSSA	Mr. Robert Webster, ESC/ENS
Ms. Nanci Hannon, DISA/JIEO	Mr. Mark Wilson, NSWC/DD/WO
Ms. Anne Quinn, SEI/DLA	Ms. Cindy Wright, DISA/CIM/TXE

Model Developers - donated information, guidance, and examples provided in Volume II, Appendices E through J, for using specific cost models to estimate reengineering efforts:

Mr. Alan Clark	Galorath Associates	SEER-SEM
Mr. John Clark	Comptek Federal Systems	RESIZE
Mr. Tony Collins	Resource Calculations	SOFTCOST-OO
Mr. Capers Jones	Software Productivity Research	CHECKPOINT
Ms. Karen McRitchie	Galorath Associates	SEER-SEM
Mr. Jim Otte	Martin Marietta PRICE Systems	PRICE-S
Mr. Doug Putnam	Quantitative Software Management	SLIM
Mr. Larry Putnam, Jr.	Quantitative Software Management	SLIM
Mr. Mike Wood		RESIZE

Other Contributors - provided guidance and instruction on specific economic issues:

Mr. Jack Berson	HQ USA Materiel Command, Alexandria
Mr. Tom Choinski	NUWC, Newport
Ms. Anne Quinn	SEI/DLA, Pittsburgh
Ms. Paula Spinner	SAF/FMCEE, Pentagon
Mr. Robert Webster	ESC/ENS, Hanscom AFB

Table K-1. SRAH Version 1.0 Review Personnel and Organizations

AIR FORCE

Reviewer	Activity	Code	Location
Col. H. Wayne Wolfe	HQ Air Mobility Command	15CSGP/CC	Scott AFB
Mr. R. Holbert	HQ Air Mobility Command	HQ AFMC	Wright Patterson AFB
Col. Frank J. Grosso Capt. Craig Lamb Mr. Larry Lang	HQ Aeronautical Systems Center	ASC/YF	Wright Patterson AFB
Col. John Franco	Air Force Office of Scientific Research	AFOSR/XP	Bolling AFB, DC
Lt. Col. Gregory W. Krafft	HQ Air Force Materiel Command	ASC/VF	Wright Patterson AFB
Mr. Ken Fehr	HQ Air Force Materiel Command	ASC/VFWO	Wright Patterson AFB
Col. Steele	C^2IPS Program Management Office	ESC/AVI	Hanscom AFB, MA
Mr. Robert A. Webster	Electronic Systems Command	ESC/ENS	Hanscom AFB, MA
L. J. Holtzblatt	MITRE	ESC/AVI	Hanscom AFB, MA
Mr. Oscar A. Goldfarb	Department of the Air Force	SAF/AQK	Washington, DC
Mr. James D. Buckner Mr. Richard Stowers	Standard Systems Center	SSI/QI	Maxwell AFB, AL
Capt. Charles J. Locascio	United States Strategic Command	STRATCOM/J65	Offutt AFB, NE
Mr. James E. Muckey Mr. Tom Schreck	United States Strategic Command	STRATCOM/J67	Offutt AFB, NE
Mr. Deane F. Bergstrom	Rome Laboratory	RL/C3CB	Griffiss AFB, NY
Ms. Cynthia A. Lynch Mr. Ross Wainwright	Phillips Laboratory	PL/VTC	Kirtland AFB, NM
Mr. Randy J. Hansel Mr. Walter H. Lipke	HQ Oklahoma City Air Logistics Center	OC-ALC/LAS	Tinker AFB, OK
Col. Terrence G. Crossey Capt. Unholz	HQ Ogden Air Logistics Center	OO-ALC/LM	Hill AFB, UT
Mr. Gregory Magavero	HQ Arnold Engineering Development Ctr	AEDC/SCT	Arnold AFB, TN
Mr. Mike Jenkins	Space & Missile Warning Systems Program	ESC/SR	Hanscom AFB, MA
Mr. David A. Johnson Ms. Muriel Philips	HQ Electronic Systems Center	ESC/SREE	Hanscom AFB, MA
Lt. Alan R. Hendrickson Mr. Mike Rogson	HQ Space & Missile Systems Center Aerospace Corporation	SMC/MEVV	Los Angeles, CA
Mr. Roland C. Sundling	HQ Space and Missile Systems Center	SMC/SCPE	Los Angeles, CA
LTC D. Stoll	HQ Space and Missile Systems Center	SMC/CL	Los Angeles, CA
LTC Alfredo Garcia	HQ Space and Missile Systems Center	SMC/CI	Los Angeles, CA

AIR FORCE (continued)

Reviewer	Activity	Code	Location
Mr. Patrick M. Stapleton	HQ Space and Missile Systems Center	SMC/CUV/TA	Vandenberg AFB, CA
Mr. Byron Nichols Lt. Jennifer Walker	HQ Space and Missile Systems Center	SMC/CZG	Los Angeles, CA
Mr. John Schnackenberg	ARINC Research Corp. for SMC	SMC/CZG	Los Angeles, CA
Mr. F. E. Goroszko Mr. K. M. O'Neill	Space & Missile System Center/MEE	SMC/MEE	LAAFB, CA
Lt. Andy Sullivan	MILSATCOM Joint Program Office	SMC/MCDE	
Dr. Edwin M. Candler Mr. Steve Kaniuga	Wright Laboratory	WL/CC	Wright Patterson AFB, OH
William L. Curtice, III Mr. Igor Golovcsenko	HQ Aeronautical Systems Center	ASC/YWE	Wright Patterson AFB, OH
Lt. Col. George W. Adair Mr. Steve Quick	HQ 648th Air Base Group	648CCSS/CC	Brooks AFB, TX
Col. James R. Rhoades Maj. Robin Sites	HQ Air Mobility Command	AMC/XPR	Scott AFB, IL
Col. John Barnhart Col. Mike Christensen Mr. Jim Allen Mr. Ben Bolton Mr. James Buckner Maj. Darden Ms. Ann Hollon Mr. Frank Lewis Mr. Dan Nieman Mr. Sid Ogletree Capt. Jeff Palmer Capt. Ramos Mr. Lester Reagan Mr. Bob Rudder 2nd Lt. Samuels Mr. Frank Spruce Mr. Lee Stanford Mr. Henry Thessen Mr. Jerry White Mr. J. T. Wright	Standard Systems Center	SSC	Maxwell AFB, Gunter Annex, AL
Mr. Chris Sittenauer	Software Technology Support Center	STSC	Hill AFB, UT
Mr. Mike Olsem	Science Applications International Corp.	STSC/SAIC	Hill AFB, UT
Maj. Michael P. Woltz	AFOTEC		

AIR FORCE (continued)

Reviewer	Activity	Code	Location
Mr. Richard Seibel		AFMPC/DPMDX	Randolph AFB, DC
Mr. Stephen J. Hannan	HQ Air Reserve Personnel Center	ARPC/SCQ	Denver, CO
Col. Alphonso Obuchowski Capt. Sadnavitch	HQ AF Medical Support Agency	USAF/SGSI	Brooks AFB, TX
Capt. Robert P. Bubello	Air Force Materiel Command	AETC/XOR-SYSREP	Randolph AFB, TX
Mr. Daniel V. Ferens	Air Force Institute of Technology	AFIT/LAS	Wright Patterson AFB, OH
Mr. Anthony M. Clark		1854 CCISS/SM	

ARMY

Reviewer	Activity	Code	Location
Mr. Jeffrey Herman Mr. Dennis J. Turner	HQ Communications-Electronics Command	AMSEL	Fort Monmouth, NJ
Mr. Jack Berson Ms. Maryann P. Dominiak Mr. Ken Freund	HQ U.S. Army Materiel Command	HQ/AMC	Alexandria, VA
Ms. Sandra Rittenhouse	HQ U.S. Army Materiel Command	HQ/AMCAQ-B	Alexandria, VA
Ms. Nancy Engle Mr. Bruce Lewis Mr. Richard A. Manley Mr. Jerry A. Nabors	U.S. Army Missile Command	MICOM	Redstone Arsenal, AL
Mr. Thomas E. Parker Mr. Leonard W. Poznaniak	Department of the Army, Information Systems	SDSLE-DOE	Letterkenny Army Depot, PA

NAVY

Reviewer	Activity	Code	Location
Cdr. C. Middlebrook R. J. Przeciak	Fleet Combat Directions Systems Support	6060	Virginia Beach, VA
Rear Adm. J. G. Hekman Mrs. Margaret Powell	Naval Information Systems Management Center	5230	Arlington, VA
Capt. R. D. Vroman Mr. Lester Hummel	Navy Fleet Materiel Support Office	NFMSO	Mechanicsburg, PA
Cdr. Larry D. Jankne Mr. Tom Coyle Mr. Tony Guido	Naval Air Systems Command HQ	NAVAIR 546	Arlington, VA
Mr. Mark Wilson	Naval Surface Warfare Center	NSWCDD/W0/B44	White Oak, MD

NAVY (continued)

Reviewer	Activity	Code	Location
Mr. Warren Axtell Mr. Thomas C. Choinski Ms. Rosemary Howbrigg Mr. James Ionata Mr. Daniel Organ Ms. Nannette Savage	Naval Undersea Warfare Center	2151	Newport, RI

MARINES

Reviewer	Activity	Code	Location
Mrs. Marvella McDill Ms. Linda D. Salisbury	MC Computer and Telecommunications	CTAAI	Quantico, VA
Ms. Susan Crosswait Mr. B. M. Doucette	MC Tactical Systems Support Activity	HQ-10B	Camp Pendleton, CA

GOVERNMENT

Reviewer	Activity	Code	Location
Ms. Joanne P. Arnette Ms. Cynthia Wright Ms. Tamra Moore	Defense Information Systems Agency	DISA/CIM	Arlington, VA
Ms. Nanci Hannon Mr. Isaac T. Jackson	Defense Information Systems Agency	DISA/JIEO/TBE	Reston, VA
Mr. Paul Wofsy	NEXRAD Joint System Program Office	NEXRAD/JSPO	Silver Springs, Md

INDUSTRY AND ACADEMIA

Reviewer	Activity	Code	Location
Ms. Anne Quinn Mr. Douglas Waugh	Software Engineering Institute	SEI	Carnegie Mellon University Pittsburgh, PA
Mr. Daniel C. Clouser	The MITRE Corporation	MITRE	McLean, VA
Mr. Ted Davis	Software Productivity Consortium	SPC	Herndon, VA
Dr. Rick Hefner Mr. Iva Voldase	TRW/System Development Division	TRW	Redondo Beach, CA
Dr. Robert Arnold	Software Evolution Technology, Inc.	SEVTEC	Herndon, VA
Mr. Donn DiNunno	Computer Sciences Corporation	CSC	Falls Church, VA
Mr. Frank A. Sigmund	Loral Defense Systems	LORAL	Akron, OH
Dr. Bill Samson	University of Abertay		Dundee, UK
Mr. Mike Wood	Quest Integrated		Kent, WA